Learned dictionaries for sparse image representation: Properties and results

Karl Skretting and Kjersti Engan

University of Stavanger, Dept. of Electrical Eng. and Computer Science, Norway

ABSTRACT

Sparse representation of images using learned dictionaries have been shown to work well for applications like image denoising, impainting, image compression, etc. In this paper dictionary properties are reviewed from a theoretical approach, and experimental results for learned dictionaries are presented. The main dictionary properties are the upper and lower frame (dictionary) bounds, and (mutual) coherence properties based on the angle between dictionary atoms. Both l_0 sparsity and l_1 sparsity are considered by using a matching pursuit method, order recursive matching Pursuit (ORMP), and a basis pursuit method, i.e. LARS or Lasso. For dictionary learning the following methods are considered: Iterative least squares (ILS-DLA or MOD), recursive least squares (RLS-DLA), K-SVD and online dictionary learning (ODL). Finally, it is shown how these properties relate to an image compression example.

Keywords: Sparse coding, dictionary learning, dictionary properties, image compression

1. INTRODUCTION

Image representation using sparse approximation and learned overcomplete dictionaries has been given some attention in recent years. Several theoretical works have been presented and possible applications are given, like face compression,[1–4] general image compression,[5,6] image denoising,[7–9] missing pixels or impainting,[10,11] texture classification[12] or discrimination.[13]

One challenge that is common to all applications that uses learned dictionaries is how to assess the dictionary. Can the dictionary quality be quantified in some way? What would be relevant quality measures? These are not easy questions to answer. Is it a good dictionary for the given task or could it be that other unknown dictionaries might perform much better? Avoiding these questions, one possible approach is simply comparing the application performance of several learned dictionaries to each other and to the performance of other (state-of-the-art) methods.

In this paper we are searching for ways to quantify the quality of a learned dictionary. In Section 2 some dictionary properties are presented. Section 3 starts by presenting these properties on a set of chosen dictionaries. This is followed by an experiment (Sec. 3.1) were RLS-DLA parameters are evaluated in terms of the defined properties. In Section 3.2 some learning methods (MOD, K-SVD, RLS-DLA, ODL) and two pursuit methods (LARS, ORMP) are combined and tested using the same training sets. The proposed properties for the different dictionaries are finally presented. Finally, in Section 3.3 the learned dictionaries are tested in a simple compression application. The results give a possibly better understanding of the learning algorithms.

We use the following notation: Matrices are represented by upper case letters, ex. D, X, vectors are represented by lower case letters, ex. x, u, scalars by Greek letters, ex. α, λ. $w(k)$ means entry k of vector w, d_i is the vector numbered i in a set of vectors or column i in matrix D and D_i is matrix D in iteration number i. $\|\cdot\|$ denotes the l_2-norm for vectors and the Frobenius norm, or trace norm, for matrices. The squared of the Frobenius norm is $\|A\|_F^2 = \text{trace}(A^T A) = \sum_i \sum_j A(i,j)^2$. $\|\cdot\|_1$ is the l_1-norm (sum of absolute values) and $\|\cdot\|_0$ is the l_0 pseudo-norm which is the number of non-zero elements.

E-mail: karl.skretting@uis.no, kjersti.engan@uis.no

1.1 Sparse approximation

In this paper we consider real finite dimensional and normalized dictionaries. The dictionary can be represented as a matrix, where the columns of the matrix are the dictionary atoms, and all dictionary atoms have unit norm:

$$D = [d_1, d_2, \ldots, d_K] \in \mathbb{R}^{N \times K}, \qquad d_i \in \mathbb{R}^N \quad \text{and} \quad ||d_i|| = 1 \quad \text{and} \quad N \leq K. \tag{1}$$

We are concerned with signal vectors x, for example pixels from a patch of an image ordered as a column vector according to the *Sparse Land* model of Elad.[3] Thus a signal vector can be approximated by a linear combination of dictionary atoms giving:

$$\tilde{x} = \sum_{k=1}^{K} w(k) d_k = Dw, \qquad r = x - \tilde{x} = x - Dw, \tag{2}$$

with a sparseness constraint on the coefficient vector w, $\|w\|_0 \leq s_0$ or $\|w\|_1 \leq s_1$. The problem of finding w is the *sparse approximation* problem and it is often formulated as follows:

$$w_{opt} = \arg\min_w \|w\|_p + \gamma \|x - Dw\|_2^2 \qquad p \in \{0, 1\}. \tag{3}$$

As γ increases the solution is getting more dense. Setting $p = 0$ gives an NP-hard problem. Good, but not necessarily optimal, solutions can be found by matching pursuit algorithms,[14,15] for example the order recursive matching pursuit[16] (ORMP) algorithm. Setting $p = 1$ in Eq. 3 gives a feasible optimization problem. The least angle regression[17] (LARS) algorithm is effective for solving this problem. Both ORMP and LARS find w in a greedy way, by gradually increasing γ and thus select more vectors in the sparse representation until a stopping criterion is reached. The stopping criterion can be that either $\|w\|_p$ or $\|x - Dw\|^2$ have reached a predefined limit. For image representation it is usually better to use the error limit $\|x - Dw\|^2 < \delta_{err}$ as stopping criterion, since a limit on $\|w\|_p$ will give a much larger error for patches with many details than for flat patches. Practical implementations of both ORMP and LARS will be more effective when a fixed dictionary D can be used to find solutions to several signal vectors.

1.2 Dictionary learning

A common setup for the dictionary learning problem[3] starts with access to a training set, $\{x_l\}_{l=1}^L$, $x_l \in \mathbb{R}^N$, and the aim is to find both a dictionary, $D \in \mathbb{R}^{N \times K}$, and a corresponding coefficient set $\{w_l\}_{l=1}^L$, $w_l \in \mathbb{R}^K$. This setup is used in both the method of optimized directions[18] **MOD**, the family of MOD algorithms is alternatively denoted iterative least squares dictionary learning algorithms[19] (ILS-DLA), and in **K-SVD**.[1] Let X denote a matrix with x_l as columns and W a matrix with the corresponding coefficients w_l as columns. The dictionary learning problem can be formulated as an optimization problem with respect to W and D:

$$\{D_{opt},\ W_{opt}\} = \arg\min_{D,W} \sum_{l=1}^{L} ||w_l||_p + \gamma \|X - DW\|^2 \quad \text{and} \quad \|d_i\| = 1 \tag{4}$$

where $\|X - DW\|^2 = \sum_{l=1}^L \|r_l\|^2$. This is a very hard optimization problem, and a practical relaxation is to split the problem into two parts, iteratively solved:

1) Keeping D fixed find W, this gives L independent problems as in Eq. 3, and

2a) MOD: Keeping W fixed find D using the least squares solution $D = (XW^T)(WW^T)^{-1}$.

2b) K-SVD: Keeping non-zero positions in W fixed and find D and W using SVD decompositions.

Part 1 is computationally most demanding, and with this setup there will be a lot of computational effort between each dictionary update, slowing down the learning process. This is especially true with large training sets, which will often be the case for image processing applications. Some minor adjustments can be done to improve the learning speed, here denoted as the **large-MOD** variant. The training set is (randomly) divided into M (equal sized) sets, $\{X_m\}_{m=1}^M$. Now, for each iteration we only use one, some few, (or perhaps almost all) of the subsets of training vectors in part 1 above. Which subsets to use may be randomly chosen, or the subsets

which has been unused for the longest time can be chosen. Anyway, finding new coefficients for only a part of the total number of training vectors reduces the time used. In part 2 the matrices A_i and B_i are calculated by adding products of training vector submatrices and coefficient submatrices:

$$A_i = \sum_m W_m W_m^T, \qquad B_i = \sum_m X_m W_m^T \qquad \text{and} \qquad D_i = B_i A_i^{-1}. \tag{5}$$

The sum can be taken over the same subsets that were used in part 1 of the same iteration. An alternative, which is quite obvious for the last iterations when the dictionary do not change very much in each iteration but which can be used for all iterations, is to take the sum in Eq. 5 over all subsets in the training set even if only some of the coefficients were updated in part 1 for this iteration. This idea was used and first published in Ref. 20 as an implementation detail. Now we recognize it, together with the LS-DLA variant in next paragraph, as a useful step in the transition from MOD to RLS-DLA.

An alternative to large-MOD, here denoted **LS-DLA**, which may be easier to implement, includes a *forgetting factor* ($0 \ll \lambda_i \leq 1$) in the algorithm. This opens the possibility of letting the training set be infinitely large. X_i can be a new subset from an infinite training set or it can reuse training vectors processed before. D_{i-1} is used solving the sparse approximation problem of Eq 3, finding the corresponding coefficients W_i. The following dictionary update step can be formulated as:

$$A_i = \lambda_i A_{i-1} + W_i W_i^T, \qquad B_i = \lambda_i B_{i-1} + X_i W_i^T \qquad \text{and} \qquad D_i = B_i A_i^{-1}. \tag{6}$$

Online dictionary learning **ODL**[21] can be explained from Eq. 6. A single training vector x_i or a mini-batch X_i is processed in each iteration. The corresponding coefficients, w_i or W_i, are found and the A_i and B_i matrices are updated according to Eq. 6 (with $\lambda_i = 1$), which in the single training vector case are $A_i = A_{i-1} + w_i w_i^T$ and $B_i = B_{i-1} + x_i w_i^T$. In ODL the computationally demanding calculation of D_i is replaced by doing one iteration of the block-coordinate descent with warm restart algorithm. A little bit simplified, each column (atom) of the dictionary is updated as

$$d_j \leftarrow d_j + (b_j - D a_j)/a_j(j), \qquad \text{for} \quad j = 1, 2, \ldots, K \tag{7}$$

where d_j, b_j, a_j are columns of the D, B_i, and A_i matrices and D here is the column by column transition of the dictionary from D_{i-1} to D_i. Note that if Eq. 7 is repeated the dictionary will converge to the least squares solution, $D = B_i A_i^{-1}$, often after just some few iterations.

The recursive least squares dictionary learning algorithm **RLS-DLA**[22] can also be found by processing just one new training vector x_i in each iteration of Eq. 6. The current dictionary D_{i-1} is used to find the corresponding weights w_i. In RLS-DLA the main improvement is that instead of using Eq. 6, with x_i for X_i and w_i for W_i, the matrix inversion lemma (Woodbury matrix identity) can be used to update $C_i = A_i^{-1}$ and D_i directly without using neither the A_i nor the B_i matrix. This gives the following simple updating rules:

$$C_i = A_i^{-1} = (\lambda_i^{-1} C_{i-1}) - \alpha u u^T \qquad \text{and} \qquad D_i = D_{i-1} + \alpha r_i u^T, \tag{8}$$

where $u = (\lambda_i^{-1} C_{i-1}) w_i$ and $\alpha = 1/(1 + w_i^T u)$, $r_i = x_i - D_{i-1} w_i$ is the representation error. Note that in these steps neither matrix inversion nor matrix by matrix multiplication is needed.

The real advantage of RLS-DLA compared to MOD and K-SVD comes with the flexibility introduced by the forgetting factor λ_i. The **Search-Then-Converge**[22] scheme is particular favourable, and the idea is to forget more quickly in the beginning then forget less as learning proceeds and we get more confidence in the quality of the dictionary. This can be done by starting with $\lambda_0 < 1$ (we denote the initial forgetting factor as λ_0 even if λ_1 is the first to be used) and slowly increasing λ_i towards 1. The update scheme and λ_0 should be chosen so that the initial dictionary is soon forgotten and convergence is obtained in a reasonable amount of iterations. In Ref. 22 it was shown that the resulting dictionaries are better with regard to sparse representation of a test data set compared with MOD or K-SVD dictionaries. Since ODL also is based on the matrices A_i and B_i in Eq. 6 a forgetting factor, and thus the Search-Then-Converge scheme, can be included with ODL the same way as in RLS-DLA. In fact, it seems that a similar scheme is an option in Mairals ODL implementation in the SPArse

Modeling Software (SPAMS) toolbox available online*, but unfortunately this did not work as expected in our experiments.

Several issues, including most implementation issues, are ignored in the brief overview of dictionary learning algorithms above. Most important is the fact that the dictionary should be normalized, not necessarily in each iteration, but often enough to keep the 2-norm of each column vector (atom) close to one. For details see the referred papers.

2. DICTIONARY PROPERTIES

Dictionaries and frames are conceptually the same. Frame is the common name used in many theoretical works,[23,24] while the term dictionary has been more used in application oriented works and especially in the sparse representation context.[3] A family of elements $\{d_i\} \subseteq H$ is called a frame for the separable Hilbert space H if there exists constants $A, B > 0$ such that

$$A||x||^2 \leq \sum_i | < x, d_i > |^2 \leq B||x||^2, \qquad \text{for all} \quad x \in H. \tag{9}$$

The indices i may be elements in any countable index set. The numbers A and B are called *frame bounds*. If $A = B$ the frame is called *tight*. Ignoring most technicalities, we simply let the dictionary be a real matrix $D \in \mathbb{R}^{N \times K}$ and assume $N \leq K$. The columns of the dictionary are assumed *normalized*. The normalization of the atoms is not an important restriction since it does not change the sparse representation capabilities of a dictionary. The frame bounds are regarded as dictionary properties A and B.

2.1 Basic properties

A dictionary is an ordinary matrix, and as such common matrix properties might be relevant as dictionary properties. The singular value decomposition (SVD) is important for matrices, and the SVD of D can be written as

$$D = U\Sigma V^T, \quad UU^T = U^TU = I_N, \; VV^T = V^TV = I_K, \quad \Sigma = \begin{bmatrix} \sigma_1 & \cdots & & & 0 \\ \vdots & \ddots & & & \vdots \\ 0 & \cdots & \sigma_N & \cdots & 0 \end{bmatrix} \tag{10}$$

The singular values of D $\{\sigma_i\}_{i=1}^N$ are real and non-negative and usually in descending order which is assumed here. They give some important properties of the dictionary: The induced 2-norm of the matrix is equal to the largest singular value and the upper frame bound is equal to the same value squared, i.e. $B = ||D||_2^2 = \sigma_1^2$, the lower frame bound is equal to the smallest non-zero eigenvalue squared ($A = \sigma_N^2$, assuming full rank of D).

Two matrices that can be defined by the dictionary are the *frame operator* $S = DD^T$ and the *Gram matrix* $G = D^TD$. The eigenvalues of these matrices can be found from the singular values of D

$$S = DD^T = U\Sigma\Sigma^TU^T = U\Lambda_NU^T, \qquad G = D^TD = V\Sigma^T\Sigma V^T = V\Lambda_KV^T \tag{11}$$

S and G have the same eigenvalues, since we have $\lambda_i = \Lambda_N(i,i) = \Lambda_K(i,i) = \Sigma(i,i)^2 = \sigma_i^2$ for $i = 1, 2, \ldots, N$. If the dictionary is tight then all eigenvalues are equal. For a normalized dictionary the sum of the eigenvalues is

$$\sum_{i=1}^N \lambda_i = \sum_{i=1}^N \sigma_i^2 = ||\Sigma||^2 = ||D||^2 = \sum_{i=1}^K ||d_i||^2 = K. \tag{12}$$

Thus for a tight normalized dictionary all eigenvalues of the frame operator are $\lambda_i = K/N$. In general dictionaries are not tight, and the λ_i-values are distributed according to $0 \leq \lambda_i \leq K$ and $\sum_i \lambda_i = K$. The distribution of the eigenvalues indicates how the dictionary atoms are distributed on the unit ball in different directions, there are more atoms pointing in the directions given by eigenvectors corresponding to large eigenvalues than in directions given by eigenvectors corresponding to small eigenvalues.

*`http://www.di.ens.fr/willow/SPAMS/`

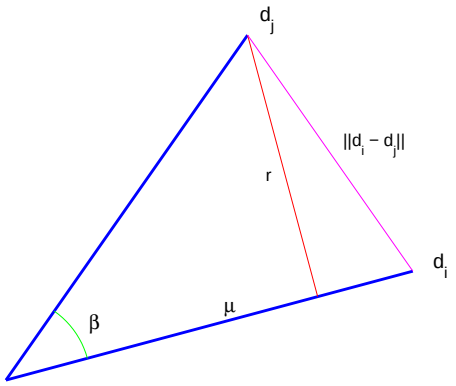


Figure 1. Different possible measures for distance or similarity between two dictionary atoms, i.e. column vectors, are shown. $||d_i|| = ||d_j|| = 1$ and $\beta \leq \pi/2$ (if $\beta > \pi/2$ then simply replace d_j by $-d_j$).

Properties concerning the spread or clustering of dictionary atoms can be important in many contexts. We have already seen that the upper frame bounds indicates the number of vectors pointing in the direction most densely represented in the dictionary and that the lower frame bound indicates the number of vectors pointing in the direction most sparsely represented in the dictionary. The distance between two normalized vectors can be measured in many ways as shown in Fig. 1. Some possible measures are:

- cosine (coherence) $\mu_{ij} = |d_i^T d_j| = \cos\beta_{ij}$ and $0 \leq \mu_{ij} \leq 1$,
- angle $\beta_{ij} = \arccos|d_i^T d_j| = \arccos\mu_{ij}$ and $0 \leq \beta_{ij} \leq \pi/2$,
- sine or representation error $r_{ij} = ||d_j - (d_i^T d_j)d_i|| = \sin\beta_{ij}$ and $0 \leq r_{ij} \leq 1$, or
- direct distance $||d_i - d_j||$.

Based on these measures, some basic dictionary properties can be defined quite naturally from how the dictionary atoms relate to each other; β_{min} is the angle between the two closest dictionary atoms, and the mutual coherence is cosine of this angle $\mu = \cos\beta_{min}$. The mutual coherence property is also used by Elad[3] and by Donoho[25] (denoted as M) among others. β_{avg} is the average angle for all atoms to its closest neighbour, and the average coherence is defined as $\mu_{avg} = \frac{1}{K}\sum_{j=1}^{K}\max_{i\neq j}|d_i^T d_j|$. Note that $\cos\beta_{avg} \neq \mu_{avg}$.

Finally, β_{mse} is the average angle of all pairs in the dictionary taken in the 'mean-squared-error" sense. The mse-properties needs some more explanation: The measure $r_{ij} = \sin\beta_{ij}$ is the error produced when atom i is represented by atom j and the squared error is $r_{ij}^2 = \sin^2\beta_{ij}$. Thus, the mean squared error is $r_{mse}^2 = \frac{1}{K^2}\sum_{i=1}^{K}\sum_{j=1}^{K} r_{ij}^2$, and the mse-properties are defined by $r_{mse} = \sqrt{r_{mse}^2} = \sin\beta_{mse}$ and $\mu_{mse} = \cos\beta_{mse}$ and

$$\begin{aligned}\mu_{mse}^2 &= 1 - r_{mse}^2 = 1 - \frac{1}{K^2}\sum_{i=1}^{K}\sum_{j=1}^{K} r_{ij}^2 = \frac{1}{K^2}\Big(K^2 - \sum_{i=1}^{K}\sum_{j=1}^{K} r_{ij}^2\Big) = \frac{1}{K^2}\sum_{i=1}^{K}\sum_{j=1}^{K}(1 - r_{ij}^2) \\ &= \frac{1}{K^2}\sum_{i=1}^{K}\sum_{j=1}^{K}\mu_{ij}^2 = \frac{1}{K^2}\sum_{i=1}^{K}\sum_{j=1}^{K}(d_i^T d_j)^2 = \frac{1}{K^2}||D^T D||^2 = \frac{1}{K^2}||\Lambda_K||^2 = \frac{1}{K^2}\sum_{i=1}^{N}\lambda_i^2. \qquad (13)\end{aligned}$$

Since $\sum_{i=1}^{N}\lambda_i = K$ it follows that μ_{mse}^2 is the sum of the eigenvalues (of $S = DD^T$) squared divided by the squared sum of the eigenvalues, i.e $\mu_{mse}^2 = (\sum_{i=1}^{N}\lambda_i^2)/(\sum_{i=1}^{N}\lambda_i)^2$. This ratio is a measure of the spread of the eigenvalues. When all eigenvalues are equal the dictionary is tight and μ_{mse} has its minimal value $\mu_{mse} = 1/\sqrt{N}$, the dictionary atoms are well spread with a large $\beta_{mse} = \arccos(1/\sqrt{N})$. The opposite case is when all dictionary atoms are the same, then one eigenvalue is K and the others are zero and μ_{mse} has its maximal value $\mu_{mse} = 1$ and the corresponding angel is small $\beta_{mse} = 0$.

The mutual coherence property μ is based on only two dictionary atoms, while the μ_{avg} property is based on K pairs, and the μ_{mse} property is based on all K^2 pairs (including an atom to itself). Any two close atoms can make μ close to one even if the rest of the atoms are well spread. μ_{avg} can be close to one even for a tight dictionary, as when the dictionary is the concatenation of two orthogonal bases and one of the bases is only a

small twist from the other. For μ_{mse} to be close to one on the other hand, the dictionary atoms must truely be clustered.

2.2 Sparse representation properties

For any signal vector $x \in \mathbb{R}^N$ and a normalized dictionary atom d_i we denote cosine of the (inner) angle between them as $\mu(d_i, x) = |d_i^T x|/||x||$. The angle between the normalized dictionary D and the vector x is now defined as the angle between the x vector and its closest dictionary column vector:

$$\mu(D, x) = \max_i \ |d_i^T x|/||x|| \qquad \text{and} \qquad \beta(D, x) = \arccos \mu(D, x). \tag{14}$$

We define β_{gap} as the angle between "the most difficult vector to represent by the dictionary" and its closest dictionary atom, the corresponding coherence is $\mu_{gap} = \cos \beta_{gap}$. The definition can be expressed as

$$\mu_{gap} = \min_{\|x\|=1} \ \max_i \ |d_i^T x| \qquad \text{and} \qquad \beta_{gap} = \arccos \mu_{gap}. \tag{15}$$

The μ_{gap} property has been used earlier, denoted as dictionary distribution ratio $B(D)$ by Liu et al.[26] and as $I(\lambda)$ by Mallat and Zhang.[27] Elad[3] defined the decay factor as $\delta(D) = \mu_{gap}^2$. We prefer the proposed name μ_{gap} since it corresponds nicely with the other defined dictionary properties, and since it is the coherence of a vector x placed in the largest "gap" of the dictionary. The worst case performance of matching pursuit algorithms can be expressed using the gap-properties. Let r^m be the matching pursuit representation error after m iterations, selecting the best atom in each iteration. This representation error will be limited by[27]

$$\|r^m\| \leq (1 - \mu_{gap}^2)^{m/2} \|x\| \qquad \text{or} \qquad \|r^m\| \leq (r_{gap})^m \|x\| \quad \text{where} \quad r_{gap} = \sin \beta_{gap}. \tag{16}$$

The ultimate test for a dictionary is to test it in a useful application, but this can hardly be called a dictionary property as it depends on many factors other than the dictionary. For a fixed set of test data, a specific sparse approximation algorithm, and a fixed sparseness s the sparse representation capabilities for a dictionary can be defined by the resulting representation error. For images this can be measured by the *peak signal to noise ratio* (PSNR). An alternative definition, which is more appropriate for images, is to measure the sparse representation capability. This can be done as follows: Define a target PSNR, `tPSNR`, and from this find an error limit: $\delta_{err} = N \cdot \texttt{peak}^2 \cdot 10^{-\texttt{tPSNR}/10}$, `peak` is usually 255. The error limit is used as a stopping criterion in a sparse approximation algorithm, i.e. $\|x - Dw\|_2^2 \leq \delta_{err}$. Finally, the *sparse representation capabilities* (`SRC`) for that dictionary is defined as the average number of non-zeros coefficients in each sparse approximation:

$$\texttt{SRC}(D, X_t, \texttt{tPSNR}) = \frac{1}{L} ||W||_0 \tag{17}$$

X_t is a test set and W the corresponding coefficient matrix found by the sparse approximation algorithm. The `SRC` value is easier to find than the more general $\mu_{gap,s}$ property.

2.3 Dictionary properties summarized

The suggested properties are summarized in Table 1. They are highly correlated and it may seem redundant to present them all. But, in line with the redundant nature of the dictionary itself, why should we not allow for some redundancy in its properties as well? Whether to use the β-properties or the μ-properties is a matter of taste. In this work we will mainly use the μ-properties.

2.4 Difference between two dictionaries

Sometimes it is interesting to quantify the similarity or difference between two dictionaries, for example when comparing a learned dictionary to a true underlying dictionary or to an initial dictionary. The measure should be independent of the order of the columns, thus the matrix norm $\|D - D'\|$ will not do. Denote the two dictionaries $D \in \mathbb{R}^{N \times K}$ and $D' \in \mathbb{R}^{N \times K'}$. The distance (similarity) can be quantified based on the angle defined in Eq. 14:

$$\beta(D, D') = \beta(D', D) = \frac{1}{K + K'} \Big(\sum_{j=1}^{K} \beta(D', d_j) + \sum_{j=1}^{K'} \beta(D, d'_j) \Big). \tag{18}$$

Property	Definition	Property	Definition
A	lower frame bound, λ_N	B	upper frame bound, λ_1
μ	$\mu = \max_{i\neq j} \lvert d_i^T d_j \rvert \quad = \cos\beta_{min}$	β_{min}	$\beta_{min} = \min_{i\neq j} \beta_{ij} \quad = \arccos\mu$
μ_{avg}	$\mu_{avg} = \frac{1}{K}\sum_{j=1}^{K} \max_{i\neq j} \lvert d_i^T d_j \rvert$	β_{avg}	$\beta_{avg} = \frac{1}{K}\sum_{j=1}^{K} \min_{i\neq j} \beta_{ij}$
μ_{mse}	$\mu_{mse} = \frac{1}{K}\sqrt{\sum_{i=1}^{N} \lambda_i^2}$	β_{mse}	$\beta_{mse} = \arccos\mu_{mse}$
μ_{gap}	$\mu_{gap} = \min_{\lvert\lvert x\rvert\rvert=1} \max_i \lvert d_i^T x \rvert$	β_{gap}	$\beta_{gap} = \arccos\mu_{gap}$

Table 1. Suggested properties for a normalized dictionary D are listed here. Note that $\mu_{avg} \neq \cos\beta_{avg}$. β_{ij} is the angle between atoms i and j, $\beta_{ij} = \arccos\mu_{ij} = \arccos\lvert d_i^T d_j \rvert$. $\lambda_i = \sigma_i^2$ is an eigenvalue (of $S = DD^T$), λ_1 is the largest one and λ_N is the smallest one, σ_i is a singular value of D as in Eq. 10. The sum of the eigenvalues is $\sum_{i=1}^{N} \lambda_i = K$.

This measure does not depend on an (optimal) pairing between the atoms in each dictionary, and K can be different from K'. We prefer to use the β-property and not a similar measure based on the coherence, since $\beta(D, D')$ behaves like other distance measures. In other words $\beta(D, D) = 0$ and the triangle property is fulfilled when D_1, D_2 and D_3 have the same size, i.e. $\beta(D_1, D_3) \leq \beta(D_1, D_2) + \beta(D_2, D_3)$.

3. EXPERIMENTS AND PROPERTIES

Three experiments were conducted to learn about, and to quantify, how the dictionaries are affected by different dictionary learning setups, including different learning methods and different choices of parameters. The main results are presented in this section, and even more details on the experiments and more results are presented on the web[†]. The first experiment uses RLS-DLA with different learning setups, the purpose is to find a suitable parameter setup as well as to learn about the effect of using l_0 versus l_1 sparsity in learning. The second experiment compares different dictionary learning methods, or variants of them, to each other. Finally, in the third experiment the learned dictionaries are used in an image compression application.

Training and test data are generated from 8 by 8 patches of the Berkeley image segmentation set[‡], only the greyscale component is used equivalent to the definition of data set 'A' in Mairal.[21] The mean is subtracted from each patch thus the spanned space is reduced to be 63-dimensional even though each vector has 64 entries. A fixed test data set, X_t of size 64×25600, is generated from randomly selected patches of the Berkeley test images. When evaluating the `SRC` measure this set is used, together with the ORMP algorithm and the stopping criterion of $\|r\|_2^2 \leq 1316$ (or mean squared error for a pixel $\approx 4.5^2$). In practice many patches will have a smaller error and the resulting PSNR will usually be somewhat larger than 36 dB.

Let D be a 64×256 matrix. To get a feeling of "typical" dictionaries in this large space, the average properties and their standard deviation for three dictionary sets as well as the properties of one specific dictionary are listed in Table 2. The first set is 100 random Gaussian dictionaries, where each entry is normally distributed, then for each atom, i.e. column vector, the mean is subtracted and the atoms are normalized to unit norm. The atoms will be uniformly distributed on the surface of a unit ball in $\mathbb{R}^{63}$. The second set consists of 100 dictionaries where each atom is a randomly chosen vector from the training set, mean subtracted and normalized to unit norm. These dictionaries are typical examples for what is used as initial dictionaries in our learning experiments. The third set consists of 10 learned dictionaries, all learned using the same setup: RLS-DLA with $\lambda_0 = 0.996$ as learning algorithm, 2 millions training vectors processed, and target PSNR set to 40. Finally, the last dictionary is the concatenation of 4 maximally separated orthogonal bases, the first 64 atoms are the standard unit vectors and all the rest of the atoms have all values equal to $\pm 1/8$.

An important choice in dictionary learning is how to do the sparse approximation, i.e. how to find the sparse representation coefficient vector w_i for each training vector x_i. It must be decided whether to use the l_0 or the l_1 sparsity in Eq. 4, and which sparse approximation algorithm and corresponding stopping criterion to use.

[†] `http://www.ux.uis.no/~karlsk/dle/`

[‡] `http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/`

Dictionary (64×256)		A	B	μ	μ_{avg}	μ_{mse}	μ_{gap}	$\beta(D, D')$	SRC
Random Gaussian	mean	1.1205	8.67	0.510	0.373	0.140	0.139	68.07°	14.95
	std	0.0675	0.25	0.027	0.003	0.000	0.005	0.14	0.10
From the images	mean	0.0844	52.15	0.961	0.672	0.288	0.046	45.30°	11.17
	std	0.0216	5.63	0.021	0.026	0.019	0.006	1.20	0.08
Learned dictionaries	mean	0.0919	16.12	0.947	0.518	0.156	0.100	39.28°	8.42
	std	0.0248	0.14	0.008	0.003	0.001	0.019	0.54	0.01
Orthogonal bases		4	4	0.125	0.125	0.125	0.125		10.87

Table 2. Properties for random Gaussian dictionaries, data dictionaries, learned dictionaries, and a dictionary which is the concatenation of 4 maximally separated orthogonal bases. Details on these dictionaries are given in the text. The $\beta(D, D')$ angle is given in degrees and measures the difference between two dictionaries of the same kind. SRC is as in Eq. 17 and the other properties are defined in Table 1.

Here we use both the l_1 norm with the LARS algorithm, and the l_0 norm with the ORMP algorithm. Fast implementations of both are available in the SPAMS software package. A target PSNR limits the representation error used as the stopping criterion for both LARS and ORMP. In general there is also a choice of which dictionary learning algorithm to use, including possible parameters, and the number of training vectors to process.

3.1 Experiment 1, RLS-DLA dictionaries

This RLS-DLA experiment uses three parameters; the initial forgetting factor, λ_0, the target PSNR and the vector selection method to use. λ_i slowly increases towards 1 in a fixed way and reaching 1 when 98% of the total number of training vectors are processed, this number also being fixed (2 millions). Both ORMP and LARS are used with target PSNR in the range from 29 to 46 and $\lambda_0 \in \{0.996, 0.997, 0.998, 0.999, 0.9995\}$. An important observation made in preliminary tests was that the dictionary properties in most cases, i.e. when λ_0 was small enough and the number of training vectors was large enough, were quite independent of the initial dictionary and the specific training vectors used during learning. The properties of the learned dictionaries mainly depend on the dictionary learning setup, i.e. the three mentioned parameters. We observed that, for a fixed choice of parameters, the properties of several learned dictionaries were almost equal, but their distances between each other, in the $\beta(D, D')$ measure, was noticeable and almost the same for any pair, see Table 2.

One dictionary was learned for each parameter combination, and the resulting B, μ_{mse}, μ_{avg} and SRC properties are shown in Fig. 2 and 3. Some variations in the dictionary properties are noticed. The properties B, μ_{mse} and μ_{avg} all gradually decrease as the target PSNR increases. The only exception is for large target PSNR and λ_0 close to 1, where the impact of the initial dictionary may still be present. Most likely because having λ_0 closer to 1 moves the dictionary more slowly away from the initial dictionary, and a large target PSNR gives small error and small update steps for the dictionary, see Eq.8. The SRC measure is calculated using ORMP and target PSNR of 35, but learning with ORMP and a target PSNR of 40 achieved the best SRC measure. Using ORMP, with increasing target PSNR, the achieved SRC decreases from 9 to its minimum of 8.4 (for PSNR = 40) and then increases to 8.6. Using LARS on the other hand, the best SRC measures were with low values for the target PSNR, within the used range the SRC measure increases from 8.5 to 8.9.

The most noticeable result of this experiment is the observation that using LARS gives dictionaries with completely different properties than ORMP, clearly seen when comparing Fig. 2 and Fig. 3, especially noting the scales on the y-axes. The essential difference between l_0 and l_1 sparsity obviously matters in dictionary learning. The dictionaries learned using LARS are close to be tight (a tight dictionary has upper frame bound $B = 4$). This may, in some applications, be a wanted property, but if the SRC measure is important the results show that learning using ORMP is a little bit better than learning using LARS.

3.2 Experiment 2, dictionaries learned by different methods

In the second experiment we want to compare different learning methods to each other. Results are presented in Table 3. Each line corresponding to different cases, and the resulting properties are averaged over 10 trials. The first line depicts the properties of dictionaries learned by MOD, and the large-MOD variant, Eq. 5, was used. In each iteration 1 to 10 random subsets (from the 50 available subsets, each with 20 thousand training vectors) were

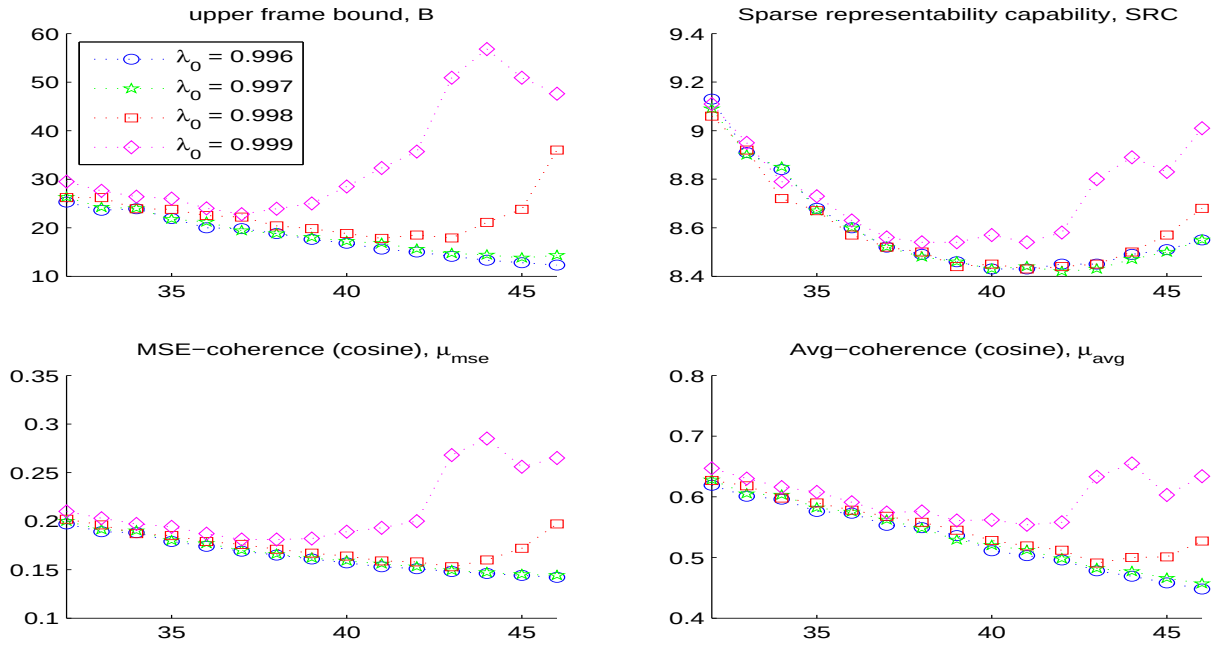


Figure 2. (Ex.1) This plot shows dictionary properties for dictionaries learned by RLS-DLA, one dotted line for each value of λ_0 equal to 0.996, 0.997, 0.998 and 0.999. Sparse approximation is done by **ORMP** and the used target PSNR is along the x-axis. See text for more information.

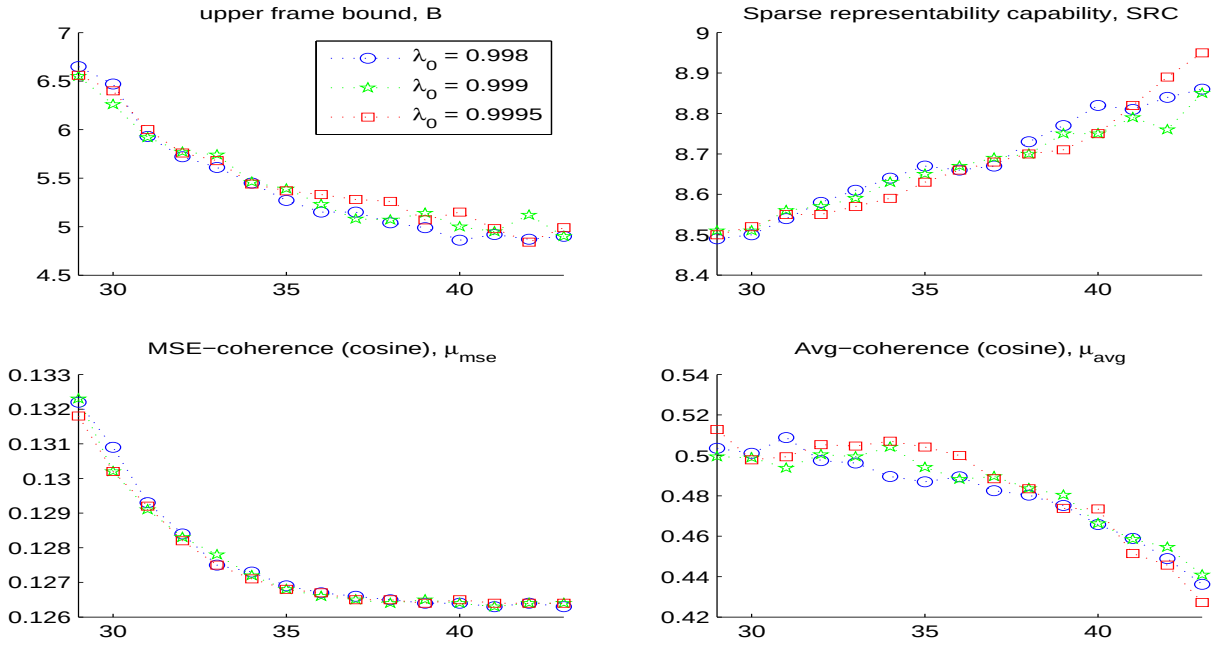


Figure 3. (Ex.1) This plot shows dictionary properties for dictionaries learned by RLS-DLA, one dotted line for each value of λ_0 equal to 0.998, 0.999 and 0.9995. Sparse approximation is done by **LARS** and the used target PSNR is along the x-axis. See text for more information.

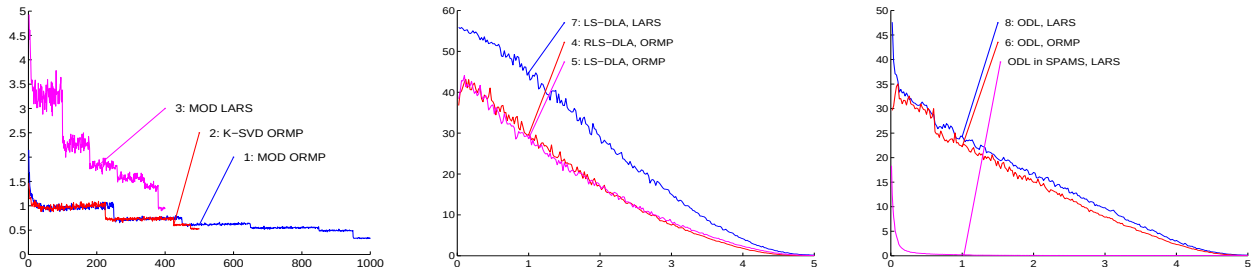
Dictionary	h:mm	Mtv	A	B	μ	μ_{avg}	μ_{mse}	μ_{gap}	$\beta(D, D')$	SRC
1. MOD, ORMP	1:26	61	0.005	33.8	0.966	0.600	0.208	0.02	33°	8.63
2. K-SVD, ORMP	5:46	18	0.002	41.7	0.967	0.624	0.238	0.01	37°	8.77
3. MOD, LARS	0:47	25	1.99	6.42	0.704	0.505	0.129	0.29	36°	8.45
4. RLS-DLA, ORMP	1:06	5	0.11	17.9	0.948	0.523	0.160	0.12	36°	8.37
5. LS-DLA, ORMP	0:18	5	0.11	17.2	0.946	0.520	0.159	0.12	36°	8.37
6. ODL, ORMP	0:24	5	0.13	17.6	0.945	0.520	0.159	0.13	35°	8.35
7. LS-DLA, LARS	0:22	5	2.11	6.22	0.697	0.500	0.129	0.28	37°	8.48
8. ODL, LARS	0:28	5	0.13	22.3	0.871	0.548	0.180	0.01	29°	8.50

Table 3. (Ex.2) Each line corresponds to different cases of learning algorithms and parameters. The resulting dictionary properties are averaged over 10 trials. `h:mm` is average time for learning and `Mtv` is millions of training vectors processed during learning. $\beta(D, D')$ is a measure for the difference between two learned dictionaries in the same set. Since the μ_{gap} properties are difficult to calculate, the given values are maximal values. A larger gap, i. e. smaller μ_{gap}, may exist in the dictionary and for the almost tight dictionaries, case 3 and 7, the μ_{gap} property should probably be below 0.125 as it is for the tight dictionary case 4 in Table 2.

used. A total of 1000 iterations were done, average learning time and the number of training vectors processed are shown in Table 3. ORMP, with a target PSNR of 38, was used for sparse approximation. The second case was similar to the first one except that K-SVD was used instead of MOD. Since K-SVD is considerable slower than MOD, at least for the Matlab implementations used here, this set was learned using 500 iterations and 1 to 4 subsets used in each. The third case was like the first, except that the sparse approximation was done by the LARS algorithm and target PSNR of 35. Here the dictionaries were learned using 400 iterations which were enough for convergence, and 1 to 10 subsets were used in each iteration. Using K-SVD with the l_1 norm in sparse approximation, i.e. LARS, makes no sense since the K-SVD dictionary update step conserves the l_0 norm but modifies the l_1 norm of the coefficients, and thus the actual coefficients will be neither l_0 nor l_1 sparse.

The next cases represent dictionaries learned by online algorithms. First, in case 4, we use the RLS-DLA as it was originally presented, i.e. using Eq. 8 and updating the dictionary for each training vector. ORMP, with target PSNR of 40, was used in the sparse approximation step, initial forgetting factor $\lambda_0 = 0.996$, and 5 millions training vectors were processed. To be able to compare RLS-DLA and ODL with each other in a setup where the two algorithms are as equal as possible, case 5 and 6 are similar to case 4, now using the mini-batch variants of LS-DLA and ODL. The mini-batch approach allows us to use the effective sparse approximation implementations in SPAMS, the size of the mini-batches increases from 50 (in the beginning) to 1000 (in the end) training vectors. Also note that the used values for λ_i in Eq. 6 are as λ_i^m (here λ_i as in Eq. 8) and m is the number of training vectors in the mini-batch. The used Matlab implementations of ODL and LS-DLA are identical except for the dictionary update step which is according to Eq. 6 for LS-DLA and Eq. 7 for ODL. Finally case 7 and 8 are similar to case 5 and 6, except for using LARS with target PSNR of 35 for sparse approximation, and larger initial forgetting factor $\lambda_0 = 0.998$.

Results are presented in Table 3. Each line corresponding to different cases, and the resulting properties are averaged over 10 trials. The standard deviation is now omitted, it is small anyway as seen in the learned dictionaries in Table 2. Studying the first two cases we see that the property values of the K-SVD dictionaries lies in between the property values of the MOD dictionaries and initial dictionaries (set 2 in Table 2), which might indicate that more iterations should be done to move the learned dictionaries further away from the initial dictionary. Fig. 4 (a) show the dictionary update step in each iteration, i.e. the $\beta(D_i, D_{i-1})$ measure. Note that for both case 1 and 2 the step sizes are small; near 1° when one subset (which is 20 thousand training vectors) is used but a little bit larger for the very first iterations, then near 0.7° when 2 subsets are used, near 0.6° when 3 subsets are used, and near 0.5° when 4 subsets are used. For the K-SVD case the number of small steps, which are not pointing in the same direction, is just not large enough to move the resulting dictionary as far away from the initial dictionary as for the MOD case. A single trial, not 10 as used in Table 3 since it used 13 hours, was done for a case with K-SVD and 1000 iterations. The resulting properties for this case was as for MOD in case 1.



(a) (b) (c)

Figure 4. (Ex.2) These plots show how much the learned dictionary changes during iterations. The y-axis is the "dictionary update step", i.e. the distance $\beta(D, D')$ in degrees, note that the scale varies. D and D' are one iteration apart for plot (a), and 20 thousand traning vectors are processed between D and D' for (b) and (c). The x-axis is iteration number, (a), or the number of training vectors processed in millions (Mtv), (b) and (c). The cases are the same as in Table 3, 1 to 3 in (a), the (R)LS cases 4, 5 and 7 in (b), and in (c) the ODL case 6 and 8 as well as the case using original ODL (SPAMS implementation) without forgetting factor (equivalent to $\lambda_i = 1$).

The properties for case 3 are quite different from the two previous cases. When changing from ORMP to LARS in MOD (case 1 and 3) we get the same effect as when changing from ORMP to LARS in RLS-DLA shown in the first experiment. The resulting dictionaries are almost tight for the LARS cases. The `SRC` measure is better for MOD-LARS than it is for MOD-ORMP, as opposed to what we see for the LS-DLA cases 5 and 7 in Table 3. Looking at dictionary update step in each iteration, Fig. 4 (a), we find a probably explanation for this. The step size is approximately three times larger when using LARS compared to ORMP. MOD, and especially K-SVD, with ORMP are not appropriate with large training set, since many (in this case ≈ 4000) iterations are needed until convergence. MOD with LARS on the other hand converged much faster, after only 400 iterations the dictionary properties were quite stable.

Comparing results for case 4 and 5 convinced us that whether to use the RLS-DLA or the LS-DLA (with small enough mini-batches) is mainly an implementation issue. Case 6 is ODL with ORMP, and the only difference from LS-DLA in case 5 is the dictionary update step. ODL does one step towards the LS-solution while LS-DLA directly calculates the LS-solution. The results in Table 3 show that the two cases give dictionaries with almost the same properties, ODL achieves a little bit better `SRC` measure. From the mathematical relation between Eq. 7 and the LS-solution it is reasonable to assume that the ODL dictionary update step is generally smaller than the LS-DLA dictionary update step, and that ODL will learn a little bit slower. This assumption is confirmed in Fig. 4; the observed update steps, i.e. $\beta(D_i, D_{i-20000})$, starts above 40° for LS-DLA in case 5 while it starts below 35° for ODL in case 6. As iterations proceed the step sizes are getting more equal to each other and after 2.5 millions training vectors they are the same for the two methods. The dip, more pronounced in the ODL cases than in the LS-DLA cases, in the step size at 0.5 millions training vectors processed is caused by an increase in the mini-batch size at this point. With the initial forgetting factor used here, $\lambda_0 = 0.996$, the ODL learning seems to have a small advantage (considering the `SRC` measure).

Finally, looking at the last two cases in Table 3 we see that LS-DLA with LARS gives a dictionary which is close to being tight, this is like MOD with LARS in case 3 and RLS-DLA with LARS in the first experiment. The results for the two cases 3 and 7 are quite similar to each other. The result for ODL with LARS case 8 is surprising. Even if the two learning algorithms of case 7 and 8 are just as closely related to each other as case 5 and 6, the results are not. ODL with LARS gives dictionaries that have completely different properties than those from the LS-DLA with LARS. We cannot explain this yet; perhaps it will be helpful to carry out a thorough investigation of dictionary development in the first iterations? It is in the first iterations that the ODL update Eq. 7 and the LS-solution differ most. The results of case 8 are more equal to the results of case 6 (and 5) than to case 7, but there are noticeable differences between case 6 and 8 as well. The dictionary update steps in Fig. 4 (c) are quite similar for cases 6 and 8, the most noticeable difference is in the very beginning where case 8 has larger update steps.

The Search-Then-Converge scheme is important to get a fast learning. Cases 4 to 8, Fig. 4 (b) and (c), all starts with large update steps as the forgetting factor is "small" in the beginning and as the forgetting factor

Dictionary	B	SRC	0.25	0.50	0.75	1.00	1.50
1. MOD, ORMP	27.9	8.56	34.32	37.75	39.79	41.25	43.52
2. K-SVD, ORMP	37.5	8.71	34.19	37.62	39.66	41.13	43.36
3. MOD, LARS	6.5	8.44	33.75	37.23	39.38	40.95	43.40
4. RLS-DLA, ORMP	18.4	8.36	34.17	37.64	39.74	41.23	43.58
5. LS-DLA, ORMP	16.8	8.35	34.15	37.61	39.70	41.21	43.57
6. ODL, ORMP	17.2	8.33	34.19	37.67	39.74	41.25	43.60
7. LS-DLA, LARS	6.3	8.46	33.52	37.04	39.21	40.82	43.31
8. ODL, LARS	22.3	8.49	34.11	37.56	39.61	41.11	43.42
D of size 64×440			34.22	37.76	39.82	41.32	43.65
D of size 64×256, learned from `lena`	31.7	10.40	34.60	38.06	39.93	41.24	43.14
DCT			34.02	37.41	39.54	41.09	43.44
JPEG 2000			35.29	38.60	40.48	41.91	44.13

Table 4. (Ex.3) Compression results for image `lena` and the dictionaries learned with different cases, as in Table 3. For each dictionary the properties B and SRC are shown, and the other columns display the achieved PSNR for bit rates from 0.25 to 1.50. The last part of the table contains results achieved with two other dictionaries, and two other methods.

increases the update steps decrease and learning moves from the search phase to the converge phase. For case 1 to 3 Fig. 4 (a) shows that the dictionary update steps are smaller, thus the dictionaries move relatively slowly away from the initial one. The step size depends on the number of subsets used (increasing in steps from 1 to 4 or 10) in each iteration, and as long as many iterations are done the dictionary may still move sufficiently away from the initial dictionary. The update steps for ODL with LARS using the SPAMS implementation without forgetting ($\lambda_i = 1$) is depicted in Fig. 4 (c) and this also shows small update steps. The dictionary change for the first 20 thousand vectors is $\approx 18°$, and quickly decreasing as learning goes on, an is only $\approx 1°$ from 180 to 200 thousand iterations, and as iterations progress continuing to decrease towards zero. Probably the dictionary does not move sufficiently away from the initial dictionary as the final difference is $\beta(D, D_0) \approx 34°$. The properties of this case, $B = 32$, $\mu_{mse} = 0.20$ and SRC= 9.6, indicate that case 8 (ODL with forgetting factor) is more favorable.

3.3 Experiment 3, image compression

In this experiment the learned dictionaries are used in an image compression application. Exactly the same compression scheme is presented in Ref. 6 and here only a brief overview is given: The image to be compressed is divided into 8 by 8 patches and arranged as column vectors in a matrix X. The mean value, the DC component, is extracted for each patch, quantized and compressed separately using a predictive method and a Huffman coder. Then the sparse matrix W is found using sparse approximation of X with the supplied dictionary D. We use ORMP with representation error as stopping criterion, and the representation error limit is adjusted to achieve different bit rates. The non-zero coefficients in W are quantized, bin size set so that the added error corresponds to a reduction of PSNR by 0.4–0.5 dB. Finally, the quantized values and the position information are entropy coded.

One learned dictionary for each of the 8 cases presented in Table 3 is used with this compression scheme. Since 10 dictionaries were learned for each method, the one with lowest SRC measure is used here, thus the particular values for B and SRC in Table 4 are not exactly as the mean values shown in Table 3. The compression results are presented in Table 4 and Fig. 5. Also included in Table 4 are results for two other dictionaries: The first one was learned using another set of training images[6] (probably more related to the `lena` image than the training images used here) and a more redundant dictionary of size is 64×440. The second one was learned using patches from the `lena` image as training set and the target PSNR was 38, the learning method was RLS-DLA with ORMP. For the sake of comparison, the results for a DCT based compression scheme is given, the quantized DCT coefficients are compressed in a way that exploits the structure in the coefficients and using the same type of entropy coder as in the dictionary compression scheme. Finally, the Matlab implementation of the JPEG 2000 compression scheme are presented.

Comparing the compression results for the 8 learned dictionaries there are no large differences. The most pronounced is that the almost tight dictionaries, 3 and 7, are not as good as the other ones, especially at low bit

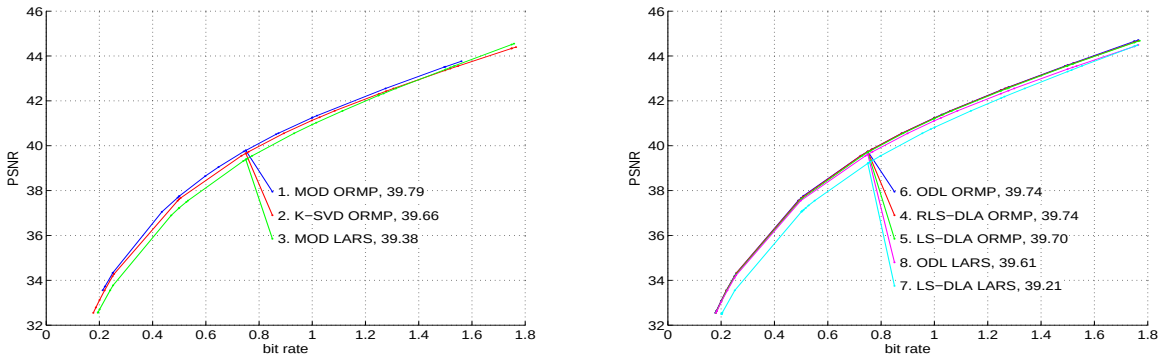


Figure 5. (Ex.3) This figure shows the compression results for different dictionaries on test image `lena`.

rates. The three dictionaries with almost the same properties (and the best `SRC` measures) from case 4, 5 and 6 achieve very similar compression results, and are best at high bit rates. For bit rates below 1 the MOD ORMP dictionary did best. It is also interesting that the dictionary learned from `lena` patches were only best for low bit rates, i.e. where the PSNR is in the neighbourhood of the target PSNR during learning.

4. CONCLUSION

In this paper, we have presented several dictionary properties which were subsequently used to characterize learned dictionaries. The dictionaries were learned using a combination of different learning algorithms, parameters and sparse approximation methods, and using a fixed dictionary size and an almost infinite set of training vectors from natural images. Especially the proposed measure for difference between two dictionaries, $\beta(D, D')$, can be helpful in dictionary learning as it can tell how much the dictionary changes during learning, both the distance to the initial dictionary and to one of the previous dictionaries.

The other properties are useful for characterizing and grouping of dictionaries. It was shown that different dictionaries learned by the same setup have properties very close to each other and that dictionaries learned by different methods, and different parameters, belonged to different groups. Even so, we were not able to find a clear correlation between any property, not even the sparse representation capability `SRC` measure, and the performance of the dictionary in an image compression application.

How useful are these dictionary properties? At this point we are not sure, but we have gained new insight during this work. For example, we were initially surprised by the differences in the "LARS-dictionaries" and "ORMP- dictionaries", however by closer inspection it makes perfectly sense. We are aware of that many new dictionary learning algorithms are presented lately, even though we have chosen to focus on a limited number. And as such it could be beneficial to compare learned dictionaries by more than just mutual coherence.

REFERENCES

[1] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," *Signal Processing, IEEE Transactions on*, vol. 54, no. 11, pp. 4311–4322, 2006.

[2] O. Bryt and M. Elad, "Compression of facial images using the K-SVD algorithm," *J. Vis. Comun. Image Represent.*, vol. 19, no. 4, pp. 270–282, 2008.

[3] M. Elad, *Sparse and Redundant Representations, from Theory to Applications in Signal and Image Processing*, Springer, New York, USA, 2010.

[4] J. Zepeda, C. Guillemot, and E. Kijak, "Image compression using the iteration-tuned and aligned dictionary," in *Proceedings ICASSP 2011*, Pargue, Czech Republic, June 2011, vol. 1, pp. 793–796.

[5] J. F. Murray and K. Kreutz-Delgado, "Learning sparse overcomplete codes for images," *The Journal of VLSI Signal Processing Systems for Signal, Image, and Video Technology*, vol. 46, no. 1, pp. 1–13, Mar. 2007.

[6] K. Skretting and K. Engan, "Image compression using learned dictionaries by RLS-DLA and compared with K-SVD," in *Proceedings ICASSP 2011*, Pargue, Czech Republic, June 2011, vol. 1, pp. 1517–1520.

[7] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *IEEE Transactions on Image Processing*, vol. 15, no. 12, pp. 3736–3745, Dec. 2006.

[8] J. Mairal, G. Sapiro, and M. Elad, "Learning multiscale sparse representations for image and video restoration," *SIAM Multiscale Modeling and Simulation*, vol. 7, no. 1, pp. 214–241, Apr. 2008.

[9] J. Mairal, M. Elad, and G. Sapiro, "Sparse representation for color image restoration," *IEEE Transactions on Image Processing*, vol. 17, no. 1, pp. 53–69, Jan. 2008.

[10] M. Aharon, M. Elad, and A. M. Bruckstein, "On the uniqueness of overcomplete dictionaries, and a practical way to retrieve them," *Linear algebra and its applications*, vol. 416, pp. 58–67, 2006.

[11] Bin Shen, Wei Hu, Yimin Zhang, and Yu-Jin Zhang, "Image inpainting via sparse representation," in *Proceedings ICASSP 2009*, Taipei, Taiwan, Apr. 2009, vol. 1, pp. 697–700, Digital object identifier: 10.1109/ICASSP.2009.4959679.

[12] K. Skretting and J. H. Husøy, "Texture classification using sparse frame based representations," *EURASIP Journal on Applied Signal Processing*, vol. 2006, 2006, Article ID 52561.

[13] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman, "Discriminative learned dictionaries for local image analysis," in *IEEE Conference on Computer Vision and Pattern Recognition*, Anchorage, Alaska, USA, 2008.

[14] S. F. Cotter, J. Adler, B. D. Rao, and K. Kreutz-Delgado, "Forward sequential algorithms for best basis selection," *IEE Proc. Vis. Image Signal Process*, vol. 146, no. 5, pp. 235–244, Oct. 1999.

[15] K. Skretting and J. H. Husøy, "Partial search vector selection for sparse signal representation," in *NORSIG-03*, Bergen, Norway, Oct. 2003, available at `http://www.ux.uis.no/~karlsk/`.

[16] M. Gharavi-Alkhansari and T. S. Huang, "A fast orthogonal matching pursuit algorithm," in *Proc. ICASSP '98*, Seattle, USA, May 1998, pp. 1389–1392.

[17] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, "Least angle regression," *Annals of Statistics*, vol. 32, pp. 407–499, 2004.

[18] K. Engan, S. O. Aase, and J. H. Husøy, "Method of optimal directions for frame design," in *Proc. ICASSP '99*, Phoenix, USA, Mar. 1999, pp. 2443–2446.

[19] K. Engan, K. Skretting, and J. H. Husøy, "A family of iterative LS-based dictionary learning algorithms, ILS-DLA, for sparse signal representation," *Digital Signal Processing*, vol. 17, pp. 32–49, Jan. 2007.

[20] K. Skretting, *Sparse Signal Representation using Overlapping Frames*, Ph.D. thesis, NTNU Trondheim and Høgskolen i Stavanger, Oct. 2002, available at `http://www.ux.uis.no/~karlsk/`.

[21] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online dictionary learning for sparse coding," in *ICML '09: Proceedings of the 26th Annual International Conference on Machine Learning*, New York, NY, USA, jun 2009, pp. 689–696, ACM.

[22] K. Skretting and K. Engan, "Recursive least squares dictionary learning algorithm," *IEEE Transactions on Signal Processing*, vol. 58, pp. 2121–2130, Apr. 2010, Digital object identifier: 10.1109/TSP.2010.2040671.

[23] R. J. Duffin and R. C. Schaeffer, "A class of nonharmonic Fourier series," *Transactions of the American Mathematical Society*, vol. 72, pp. 341–366, 1952.

[24] I. Daubechies, *Ten Lectures on Wavelets*, Society for Industrial and Applied Mathematics, Philadelphia, USA, 1992, Notes from the 1990 CBMS-NSF Conference on Wavelets and Applications at Lowell, MA.

[25] D. L. Donoho and M. Elad, "On the stability of the basis pursuit in the presence of noise," *Signal Processing*, vol. 86, pp. 511–532, Aug. 2006.

[26] Qiangsheng Liu, Qiao Wang, and Lenan Wu, "Size of the dictionary in matching pursuit algorithm," *IEEE Transactions on Signal Processing*, vol. 52, no. 12, pp. 3403–3408, Dec. 2004.

[27] S. G. Mallat and Z. Zhang, "Matching pursuit with time-frequency dictionaries," *IEEE Trans. Signal Processing*, vol. 41, no. 12, pp. 3397–3415, Dec. 1993.