Universitetet
i Stavanger

Det teknisk-
naturvitenskapelige fakultet

Stavanger, April 13, 2010

Recursive Least Squares Dictionary Learning Algorithm

Karl Skretting and Kjersti Engan

This work was first printed in IEEE Trans. Signal Process. vol 58, no. 4, April 2010. It is without the IEEE layout and some minor changes, i.e. small comments, are added as footnotes.

Abstract

We present the Recursive Least Squares Dictionary Learning Algorithm, RLS-DLA, which can be used for learning overcomplete dictionaries for sparse signal representation. Most Dictionary Learning Algorithms presented earlier, for example ILS-DLA and K-SVD, update the dictionary after a batch of training vectors has been processed, usually using the whole set of training vectors as one batch. The training set is used iteratively to gradually improve the dictionary. The approach in RLS-DLA is a continuous update of the dictionary as each training vector is being processed. The core of the algorithm is compact and can be effectively implemented.

The algorithm is derived very much along the same path as the recursive least squares (RLS) algorithm for adaptive filtering. Thus, as in RLS, a forgetting factor λ can be introduced and easily implemented in the algorithm. Adjusting λ in an appropriate way makes the algorithm less dependent on the initial dictionary and it improves both convergence properties of RLS-DLA as well as the representation ability of the resulting dictionary.

Two sets of experiments are done to test different methods for learning dictionaries. The goal of the first set is to explore some basic properties of the algorithm in a simple setup, and for the second set it is the reconstruction of a true underlying dictionary. The first experiment confirms the conjectural properties from the derivation part, while the second demonstrates excellent performance.

Karl Skretting, Institutt for data- og elektroteknikk (IDE), University of Stavanger, 4036 Stavanger.
Switch board 51 83 10 00. Direct 51 83 20 16. E-mail: karl.skretting@uis.no.

1 Introduction

Signal representation is a core element of signal processing, as most signal processing tasks rely on an appropriate signal representation. For many tasks, i.e. compression, noise removal, communication, and more, a *sparse* signal representation is the preferred one. The sparseness can be achieved by thresholding the coefficients of an orthogonal basis, like the discrete cosine transform, or a wavelet basis. Also, overcomplete dictionaries are well suited for sparse representations or approximations.

The approximation $\tilde{x} = \sum_k w(k) d^{(k)}$ is formed as a linear combinations of a set of *atoms* $\{d^{(k)}\}_k$ which constitutes the dictionary. When most of the coefficients $w(k)$ are zero the approximation is sparse. General dictionaries, appropriate for a wide range of signals, can be designed in many ways, for example by scaling and translation of some basis functions as for Gabor and wavelet frames. Specialized dictionaries, intended to be used with a particular class of signals, can be *learnt* from a large set of signals reflecting the class. Learning is often restricted to the process of training a dictionary based on a set of training data, whereas design is used in a broader context including dictionary size and structure and implementation issues. Several dictionary learning algorithms (DLAs) have been presented the recent years [1]-[8]. It has been observed that there are many similarities between the dictionary learning problem and the vector quantizing (VQ) codebook design problem, which also lead to similarities in the algorithms.

Our previous work on dictionary learning started with the Method of Optimized Directions (MOD) [9, 10]. In MOD dictionary update is the Least Squares (LS) solution to an overdetermined system of equations [11, 12]. Several variants of MOD were also presented [13, 14]. The essence of these works is summarized in [5] where the LS approach of the different variants is clearly presented, and included in the name of this family of algorithms, Iterative Least Squares Dictionary Learning Algorithm, ILS-DLA. In this paper we go one step further and develop the algorithm into a Recursive Least Squares (RLS) algorithm in a similar way as ordinary LS can be developed into RLS in the adaptive filter context.

The k-means method, and the many variants and additions of it, have been used for classification, data clustering and VQ codebook design since the introduction several decades ago. In k-means a set of class centers, called a *codebook*, which here corresponds to the dictionary, is learnt from a larger set of training data. The k-means variants may be divided into two main classes, *batch* and *continuous*, depending on how often the codebook is updated. The batch-methods, i.e. the (Generalized) Lloyd Algorithm (GLA) which is also known as the Linde-Buzo-Gray (LBG) algorithm, update the codebook only after (re)-classification of all the training vectors in the training set [15]-[18]. The continuous-method, the MacQueen variant [19], updates the codebook

after classification of each training vector. Both variants have been widely referred, cited, refined, analyzed and used, but the GLA is perhaps easier to understand and analyze and thus has been more used. The reference to k-means is highly relevant when explaining how the new method, RLS-DLA, relates to other methods, especially ILS-DLA. In short: Like ILS-DLA reduces to the batch variant of k-means when used for VQ codebook design, RLS-DLA reduces to the continuous variant of k-means.

The outline of this paper is as follows: First, a brief introduction to dictionary learning is given in Sec. 1.1. The new algorithm, RLS-DLA, is presented in Sec. 2. Finally, in Sec. 3, experiments comparing RLS-DLA to ILS-DLA and K-SVD demonstrate the good performance of RLS-DLA.

We use the following notation: matrices are represented by large letters, ex. D, X, vectors are represented by small letters, ex. x, u, scalars by Greek letters, ex. α, λ. $w(k)$ is entry k of vector w, $d^{(k)}$ is the vector numbered k in a set of vectors or column k in matrix D, and D_i is matrix D in iteration number i. $\|\cdot\|$ denotes the 2-norm for vectors and the Frobenius norm, or trace norm, for matrices. The squared of the Frobenius norm is $\|A\|_F^2 = \text{trace}(A^T A) = \sum_i \sum_j A(i,j)^2$.

1.1 Dictionary learning

A dictionary is a collection of *atoms* and can be used for signal representation. The representation is a linear combination of some of the atoms in the dictionary, it can be exact or it can be an approximation to the original signal. A dictionary and a *frame* is often regarded as the same thing, but the (tiny) difference is that a frame spans the signal space while a dictionary does not have to do this.

Here, the original signal is represented as a real column vector x of finite length N and we have a finite number K of dictionary atoms $\{d^{(k)}\}_{k=1}^K$ which also have length N. The dictionary is represented as a matrix $D \in \mathbb{R}^{N \times K}$, where each atom is a column in D. The representation, or the approximation, and the representation error can be expressed:

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

where $w(k)$ is the coefficient used for atom k. The coefficients, also called weights, are collected into a column vector w of length K. In a *sparse* representation only a small number s of the coefficients are allowed to be non-zero. This gives the *sparseness factor* as s/N.

Dictionary learning is the task of learning or training a dictionary such that it is well adapted to its purpose. This is done based on some available training

data, here a set of training vectors, representing a signal class. Usually the objective is to give a sparse representation of the training set, making the total error as small as possible, i.e. minimizing the sum of squared errors. Let the training vectors constitute the columns in a matrix X and the sparse coefficient vectors the columns in a matrix W. Using the cost function $f(\cdot)$, this can be stated formally as a minimization problem

$$\begin{aligned} \arg\min_{D,W} f(D,W) &= \arg\min_{D,W} \sum_i ||r_i||_2^2 \\ &= \arg\min_{D,W} \|X - DW\|_F^2 \end{aligned} \tag{2}$$

where there is imposed a sparseness criterion on W, i.e. the number of non-zero elements in each column is given as s. The reconstruction matrix is $\tilde{X} = DW$, and the reconstruction error is $R = X - \tilde{X} = X - DW$.

A practical optimization strategy, not necessarily leading to a global optimum, can be found by splitting the problem into two parts which are alternately solved within an iterative loop. The two parts are:

1) Keeping D fixed, find W.
2) Keeping W fixed, find D.

This is the same strategy used in GLA, in ILS-DLA [5], and partly also in K-SVD [4], but not in our new algorithm RLS-DLA. Convergence of this method is problematic.

In the first part, where D is fixed, the weights are found according to some rules, i.e. a sparseness constraint. The vector selection problem,

$$w_{opt} = \arg\min_w \|x - Dw\|_2 \qquad \text{s.t.} \quad \|w\|_0 \leq s, \tag{3}$$

is known to be NP-hard [20]. The psuedo-norm $\|\cdot\|_0$ is the number of non-zero elements. A practical approach imply that the coefficients w must be found by a vector selection algorithm, giving a good but not necessarily the optimal solution. This can be a Matching Pursuit algorithm [21, 22], like Basic Matching Pursuit (BMP) [23], Orthogonal Matching Pursuit (OMP) [20, 24], or Order Recursive Matching Pursuit (ORMP) [25], or other vector selection algorithms like FOCUSS [26, 27] or Method of Frames [28] with thresholding. We use ORMP in this work since it gives better results than OMP [21], and the complexity, with the effective QR implementation, is the same [22].

Part 2 above, i.e. finding D when W is fixed, is easier. This is also where ILS-DLA and K-SVD are different. ILS-DLA uses the least squares solution, i.e. the dictionary that minimizes an objective function $f(D) = \|X - DW\|_F^2$ as expressed in (2) when W is fixed. The solution is

$$D = (XW^T)(WW^T)^{-1}. \tag{4}$$

K-SVD use another approach for part 2. The problem is slightly modified by also allowing the values of W to be updated but keeping the positions of the

non-zero entries fixed. For each column k of D a reduced SVD-decomposition, returning only the largest singular value σ_1 and the corresponding singular vectors u_1 (left) and v_1 (right), of a matrix $E' = X' - D'W'$ is done. To explain the marked matrices let I_k be the index set of the non-zero entries of row k in W, then in a MATLAB-like notation $X' = X(:, I_k)$ and $W' = W(:, I_k)$ and $D' = D$ but setting column k to zeros, i.e. $D'(:, k) = 0$. After SVD is done column k of D is set as $d^{(k)} = D(:, k) = u_1$ and the non-zero entries in the corresponding row of W is set as $W(k, I_k) = \sigma_1 v_1^T$. This procedure is repeated for all columns of D.

The least squares minimization of (2) and the sparseness measure of (3) are only two of many possible dictionary learning approaches available. Other problem formulations and objective functions can be chosen, leading to other dictionary learning algorithms [7, 29, 8]. It is especially advantageous to use the 1-norm as a sparseness measure since vector selection then can be done in an optimal way. [1]

2 Recursive Least Squares Dictionary Learning Algorithm

This section starts with the main derivation of RLS-DLA in Sec. 2.1. The resulting algorithm can be viewed as a generalization of the continuous k-means algorithm, the MacQueen variant. The derivation follows the same lines as the derivation of RLS for adaptive filter, thus the name RLS-DLA. Furthermore, a *forgetting factor* λ is included in the algorithm in Sec. 2.2, similar to how the forgetting factor is included in the general RLS algorithm. In Sec. 2.3 the forgetting factor is used to improve the convergence properties of the algorithm. The compact core of the algorithm is presented in a simple form in Sec. 2.4. Finally, some remarks on complexity and convergence are included in Sec. 2.5 and Sec. 2.6.

2.1 Derivation

The derivation of this new algorithm starts with the minimization problem of (2) and the LS-solution of (4). Our goal is to continuously calculate this solution as each new training vector is presented. For the derivation of the algorithm we assume that we have the solution for the first i training vectors, then our task is to find the new solution when the next training vector is included.

[1] RLS-DLA is independent of the vector selection method used and works well also with 1-norm sparse coding algorithms, for example the LARS-Lasso algorithm.

The matrices are given a subscript indicating the "time step". This gives $X_i = [x_1, x_2, \ldots, x_i]$ of size $N \times i$ and $W_i = [w_1, w_2, \ldots, w_i]$ of size $K \times i$. The LS-solution for the dictionary for these i training vectors is

$$D_i = (X_i W_i^T)(W_i W_i^T)^{-1} = B_i C_i. \tag{5}$$

where

$$B_i = (X_i W_i^T) = \sum_{j=1}^{i} x_j w_j^T \tag{6}$$

$$C_i^{-1} = (W_i W_i^T) = \sum_{j=1}^{i} w_j w_j^T \tag{7}$$

Let us now picture the situation that a new training vector $x = x_{i+1}$ is made available. For notational convenience the time subscript is omitted for vectors. The corresponding coefficient vector $w = w_{i+1}$ may be given or, more likely, it must be found using the current dictionary D_i and a vector selection algorithm. The approximation and the reconstruction error for the new vector x are

$$\tilde{x} = D_i w = B_i C_i w, \qquad r = x - \tilde{x}$$

Taking into account the new training vector we can find a new dictionary $D_{i+1} = B_{i+1} C_{i+1}$ where

$$B_{i+1} = B_i + x w^T \quad \text{and} \quad C_{i+1}^{-1} = C_i^{-1} + w w^T.$$

C_{i+1} is found using the matrix inversion lemma (Woodbury matrix identity)

$$C_{i+1} = C_i - \frac{C_i w w^T C_i}{w^T C_i w + 1} \tag{8}$$

which gives

$$\begin{aligned} D_{i+1} &= (B_i + x w^T)(C_i - \frac{C_i w w^T C_i}{w^T C_i w + 1}) \\ &= B_i C_i - B_i \frac{C_i w w^T C_i}{w^T C_i w + 1} \\ &+ x w^T C_i - x w^T \frac{C_i w w^T C_i}{w^T C_i w + 1} \end{aligned} \tag{9}$$

Note the fact that C_i is symmetric, $C_i = C_i^T$, since C_i^{-1} is symmetric by definition in (7). [2]

[2] C_i and C_i^{-1} are also positive definite.

We define vector u and scalar α as

$$\begin{aligned} u &= C_i w, \qquad u^T = w^T C_i && (10) \\ \alpha &= \frac{1}{1 + w^T C_i w} = \frac{1}{1 + w^T u} && (11) \\ 1 - \alpha &= \frac{w^T u}{1 + w^T u} = \frac{w^T C_i w}{1 + w^T C_i w} \end{aligned}$$

(8) and (9) are now written

$$\begin{aligned} C_{i+1} &= C_i - \alpha u u^T. && (12) \\ D_{i+1} &= D_i - \alpha \tilde{x} u^T + x u^T - (1 - \alpha) x u^T \\ &= D_i + \alpha r u^T. && (13) \end{aligned}$$

Continuous, i.e. recursive, update of the dictionary thus only require addition of matrices (matrix and vector outer product) and a few matrix-by-vector multiplications, some even involving sparse vectors. Unlikely as it might seem from looking at (5), no matrix-by-matrix multiplication or matrix inversion is needed.

In the case of extreme sparsity, i.e. restricting each weight vector w_i to have only one non-zero coefficient and set it equal to 1, the RLS-DLA is identical to the continuous k-means algorithm as MacQueen derived it [19]. In k-means each class center is the running average of all training vectors assigned to this class. Let n_k be the number of training vectors assigned to class k after the first i training vectors are processed. A new training vector x is assigned to class k, the weight vector w is all zeros except for element k which is one. The running average update for column k in the dictionary, denoted $d^{(k)}$, is as in the MacQueen paper, $d_{i+1}^{(k)} = (n_k d_i^{(k)} + x)/(n_k + 1)$. With $x = d_i^{(k)} + r$ this can be written as

$$D_{i+1} = D_i + (r w^T)/(n_k + 1). \qquad (14)$$

The RLS-DLA update of the dictionary (13) can be written as

$$\begin{aligned} D_{i+1} &= D_i + r w^T (\alpha C_i) && (15) \\ &= D_i + (r w^T) C_i / (1 + w^T C_i w). \end{aligned}$$

The similarity between (14) and (15) is obvious, (15) reduces to (14) for the extreme sparsity case. C_i, according to definition in (7), is a diagonal matrix where entry k is $1/n_k$ and entry k of $\big(C_i/(1 + w^T C_i w)\big)$ is $1/(n_k + 1)$.

2.2 The forgetting factor λ

The derivation of the RLS-DLA with a forgetting factor λ follows the same path as in the previous subsection. The dictionary that minimize the cost

function of (2) when the weight matrix W is fixed is given in (4), and for the first i training vectors in (5).

Introducing λ we let the cost function be a weighted sum of the least squares errors,

$$f(D) = \sum_{j=1}^{i} \lambda^{i-j} \|r_j\|_2^2. \tag{16}$$

where $0 < \lambda \leq 1$ is an exponential weighting factor.

As before a new training vector x (that is x_{i+1}) and the belonging coefficients w (w_{i+1}) are made available. The new dictionary D_{i+1} becomes

$$D_{i+1} = \arg\min_D f(D) = \arg\min_D \sum_{j=1}^{i+1} \lambda^{i+1-j} \|r_j\|_2^2$$

$$= \arg\min_D \left(\lambda \|\hat{X}_i - D\hat{W}_i\|_F^2 + \|x - Dw\|_2^2 \right)$$

The matrices $\hat{X}_i$ and $\hat{W}_i$ are scaled versions of the corresponding matrices in previous subsection. They can be defined recursively

$$\hat{X}_i = [\sqrt{\lambda}\hat{X}_{i-1}, x_i], \quad \hat{X}_1 = x_1$$

$$\hat{W}_i = [\sqrt{\lambda}\hat{W}_{i-1}, w_i], \quad \hat{W}_1 = w_1.$$

The new dictionary can be expressed by the matrices B_{i+1} and C_{i+1} which here are defined almost as in (6) and (7), the difference being that the scaled matrices $\hat{X}_i$ and $\hat{W}_i$ are used instead of the ordinary ones. Thus also B_{i+1} and C_{i+1} can be defined recursively

$$\begin{aligned} B_{i+1} &= \hat{X}_{i+1}\hat{W}_{i+1}^T = [\sqrt{\lambda}\hat{X}_i, x][\sqrt{\lambda}\hat{W}_i, w]^T \\ &= \lambda \hat{X}_i \hat{W}_i^T + xw^T = \lambda B_i + xw^T \\ C_{i+1}^{-1} &= \hat{W}_{i+1}\hat{W}_{i+1}^T = [\sqrt{\lambda}\hat{W}_i, w][\sqrt{\lambda}\hat{W}_i, w]^T \\ &= \lambda \hat{W}_i \hat{W}_i^T + ww^T = \lambda C_i^{-1} + ww^T \end{aligned}$$

The matrix inversion lemma gives

$$C_{i+1} = \lambda^{-1} C_i - \frac{\lambda^{-1} C_i w w^T \lambda^{-1} C_i}{w^T \lambda^{-1} C_i w + 1} \tag{17}$$

and almost as in previous subsection this gives

$$\begin{aligned} D_{i+1} &= B_{i+1}C_{i+1} = \left(\lambda B_i + xw^T\right) \cdot \\ &\quad \left(\lambda^{-1} C_i - \frac{\lambda^{-1} C_i w w^T \lambda^{-1} C_i}{w^T \lambda^{-1} C_i w + 1}\right) \\ &= B_i C_i - \lambda B_i \frac{\lambda^{-1} C_i w w^T \lambda^{-1} C_i}{w^T \lambda^{-1} C_i w + 1} \\ &+ \quad xw^T \lambda^{-1} C_i - xw^T \frac{\lambda^{-1} C_i w w^T \lambda^{-1} C_i}{w^T \lambda^{-1} C_i w + 1}. \end{aligned}$$

Here, the vector u and the scalar α are defined as

$$\begin{aligned} u &= (\lambda^{-1}C_i)w, \qquad u^T = w^T(\lambda^{-1}C_i) \\ \alpha &= \frac{1}{1 + w^T(\lambda^{-1}C_i)w} = \frac{1}{1 + w^T u} \\ 1 - \alpha &= \frac{w^T u}{1 + w^T u} = \frac{w^T(\lambda^{-1}C_i)w}{1 + w^T(\lambda^{-1}C_i)w} \end{aligned}$$

which gives

$$C_{i+1} = (\lambda^{-1}C_i) - \alpha u u^T, \tag{18}$$

$$D_{i+1} = D_i + \alpha r u^T. \tag{19}$$

This is very close to (12) and (13). The only difference is that we here use $(\lambda^{-1}C_i)$ instead of C_i.

2.3 Search-Then-Converge scheme

One of the larger challenges with k-means, ILS-DLA and K-SVD is to find a good initial dictionary. The problem seems to be that the algorithms quite often get stuck in a local minimum close to the initial dictionary. Several approaches have been used to find the initial dictionary in a good way. For the (continuous) k-means algorithm a Search-Then-Converge scheme has been used with good results [30, 31, 32]. A similar approach can also easily be adapted to the RLS-DLA, resulting in some useful hints for how to use the forgetting factor λ.

In the RLS-DLA, as derived in Sec. 2.2, it is straight forward to let λ get a new value in each step, say λ_i for step i. We ask ourselves: How can we set a good value for λ_i in step i? The Search-Then-Converge idea is to start by a small value of λ_i to quickly forget the random, and presumably not very good, initial dictionary. As the iterations proceed, we want to converge towards a finite dictionary by gradually increasing λ_i towards 1 and thus remember more and more of what is already learnt. Optimizing an adaptive λ is difficult if not impossible. Many assumptions must be made, and qualified guessing might be as good a method as anything.

A fixed forgetting factor $\lambda < 1$ effectively limits the number of training vectors on which the cost function is minimized. The finite sum of all the powers of λ in (16) is smaller than taking the sum to infinity, i.e. smaller than $\sum_{i=0}^{\infty} \lambda^i = \frac{1}{1-\lambda}$ when $|\lambda| < 1$. Setting $\lambda = 1 - 1/L'$ gives a slightly smaller total weight for the errors than the LS-solution ($\lambda = 1$) including L' error vectors. We may say that the weights of the previous dictionary and matrix C are as if they were calculated based on $L' = 1/(1-\lambda)$ previous errors. Considering the numerical

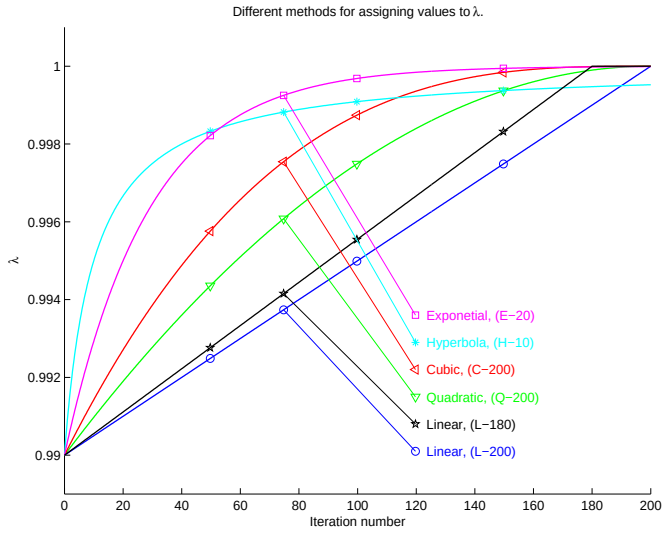

Figure 1: λ_i is plotted as a function of iteration number for the different proposed methods of increasing λ. Here one iteration is assumed to process all training vectors in the training set, to have one iteration for each training vector the x-axis numbers should be multiplied by the number of training vectors.

stability of (7), it seems reasonable not to let the horizon L' be too short. One example may be to start with $L' = 100$ and $\lambda_0 = 0.99$.

There are many functions that increase λ from λ_0 to 1, some are shown in Fig. 1. The linear, quadratic and cubic functions all set λ according to

$$\lambda_i = \begin{cases} 1 - (1-\lambda_0)(1 - i/a)^p & \text{if } i \leq a \\ 1 & \text{if } i > a \end{cases} \tag{20}$$

Parameter p is the power, and is 1 for the linear approach, 2 for the quadratic approach, and 3 for the cubic approach. The hyperbola and exponential approach set λ according to

$$\lambda_i = 1 - (1-\lambda_0)\frac{1}{1+i/a} \tag{21}$$

$$\lambda_i = 1 - (1-\lambda_0)(1/2)^{i/a}. \tag{22}$$

where the parameter a here is the argument value for which $\lambda = (\lambda_0 + 1)/2$.

2.4 The RLS-DLA algorithm

Let D_0 denote an initial dictionary, and C_0 an initial C matrix, possibly the identity matrix. The forgetting factor in each iteration is $0 \ll \lambda_i \leq 1$. The core of the algorithm can be summarized into some simple steps. The iteration number is indicated by subscript i.

1. Get the new training vector x_i.
2. Find w_i, typically by using D_{i-1} and a vector selection algorithm.
3. Find the representation error $r = x_i - D_{i-1}w_i$.
4. Apply λ_i by setting $C^*_{i-1} = \lambda_i^{-1} C_{i-1}$.
5. Calculate the vector $u = C^*_{i-1} w_i$,
   and if step 9 is done $v = D^T_{i-1} r$.
6. Calculate the scalar $\alpha = 1/(1 + w_i^T u)$.
7. Update the dictionary $D_i = D_{i-1} + \alpha r u^T$.
8. Update the C-matrix for the next iteration,
   $C_i = C^*_{i-1} - \alpha u u^T$.
9. If needed update the matrix $(D_i^T D_i)$.
10. If wanted normalize the dictionary.

We use normalized versions of the first K training vectors as the initial dictionary D_0, but other initialization strategies are possible. The non-zero entry in each coefficient vector will be $w_i(i) = \|x_i\|_2$ for $i = 1, \ldots, K$. The C_0 matrix, according to definition in (7), will be diagonal with entry i equal to $1/\|x_i\|_2^2$.

2.4.1 Remark on step 9

Some vector selection algorithms are more effective if the inner products of the dictionary atoms are precalculated, here they are elements of the matrix $(D^T D)$. Effective implementations of OMP and ORMP both need to calculate sK inner products, where s is the number of vectors to select. Updating all inner products, after the update of D as in step 7, may be less demanding than to calculate the sK needed inner products. Suppose we have the inner product matrix for the previous iteration, $(D^T_{i-1} D_{i-1})$, then we find for $(D_i^T D_i)$

$$D_i^T D_i = (D_{i-1} + \alpha r u^T)^T (D_{i-1} + \alpha r u^T)$$

$$= D^T_{i-1} D_{i-1} + \alpha\big(v u^T + u v^T\big) + \alpha^2 \|r\|_2^2 (u u^T)$$

where $v = D^T_{i-1} r$ and $\|r\|_2^2 = < r, r > = r^T r$.

2.4.2 Remark on step 10

The norm of the dictionary atoms can be chosen freely, since the dictionary atoms may be scaled by any factor without changing the signal representation of (1), as long as the corresponding coefficients are scaled by the inverse factor. It is common to let the atoms have unit norm. The dictionary update in step 7 in RLS-DLA does not preserve the norm of the atoms, thus making rescaling necessary to keep the dictionary normalized. Since normalization constitutes a large part of the computations in each iteration, and since it is not strictly demanded, it can be skipped completely or done only for some iterations.

Before normalization a diagonal scaling matrix G must be calculated, each element on the diagonal is equal to the inverse of the 2-norm of the corresponding column of D. The normalized (subscript n) dictionary is then

$$D_n = DG.$$

To keep the approximation as well as the error unchanged, we also need to recalculate the accumulated weights of C correspondingly. Setting $W_n = G^{-1}W$ conserves the representation $\tilde{X} = DW = DGG^{-1}W = D_nW_n$. This gives

$$C_n = (W_nW_n^T)^{-1} = GCG.$$

If used, also the inner product matrix (D^TD) needs to be recalculated

$$(D_n^TD_n) = (DG)^TDG = G(D^TD)G.$$

2.5 Comparison of RLS-DLA to ILS-DLA and K-SVD

Both ILS-DLA and K-SVD use a finite training set, consisting of L training vectors. Starting with an initial dictionary $D_0 \in \mathbb{R}^{N\times K}$ and main iteration number $j = 1$, the algorithms have the following steps:

A. for $i = 1, 2, \ldots, L$

A.1. Get the training vector x_i

A.2. Find w_i using D_{j-1} and store it as column i in W.

B.1. Find D_j. ILS-DLA by (4). K-SVD by the procedure outlined in Sec 1.1.

B.2. For ILS-DLA, normalize the dictionary.

B.3. Increase j and go to A)

Loop A is equivalent to steps 1 and 2 in RLS-DLA. The dictionary update in RLS-DLA is done continuously in the inner loop, thus the extensive dictionary update of B.1 is not present i RLS-DLA. The complexity of the problem of equation (2) is dependent on the sizes N, K, and L and the sparseness $s = \|w_i\|_0$. We have $s < N < K << L$. The complexity depends on how the external parts are implemented. Using ORMP (or OMP), with QR-decomposition and pre-calculated inner products, as vector selection algorithm, its complexity is $O((s^2+N)K)$ [22]. This gives the complexity for step 2 in Sec. 2.4 and step A.2 above. Also, we assume that reduced SVD can be done with complexity proportional to the size of the input matrix, this gives complexity for the B.1 step i K-SVD as $O(sNL)$. Step B.1 for ILS-DLA includes forming the matrices B_L (6) and C_L^{-1} (7) with complexity $O((s^2 + sN)L)$ and then find D (5) $O(K^3)$. Step B.2 is O(NK), which gives $O(K^3 + (s^2 + sN)L)$ for step B. In RLS-DLA Sec. 2.4 steps 4, 8, 9 and 10 are $O(K^2)$.

Summing up: One complete iteration through the training set is $O((s^2 + N + K)KL)$ for RLS-DLA compared with $O(K^3 + (s^2 + N)KL)$ for ILS-DLA and $O((s^2 + N)KL)$ for K-SVD. Vector selection is the dominating step, except the rare case that $K^3 > NKL$. In the more common cases where $K \propto N$ and s is small ($s^2 \propto N$) all three alogrithms are $O(K^2L)$. The hidden constants in these complexity expressions are important, so the actual running times may be more useful for comparing algorithm speed. In the experiment in Sec. 3.1 we have $s = 4$, $N = 16$ $K = 32$ and $L = 4000$ and the observed running times for one iteration through the training set were approximately 0.05 seconds for ILS-DLA, 0.20 seconds for RLS-DLA and 0.43 seconds for K-SVD. The implementation uses a combination of MATLAB functions and Java classes, and the experiments were executed on a PC.

2.6 Reflections on convergence

A complete analysis of the convergence properties of the algorithm is difficult. The VQ-simplification of RLS-DLA was proved to converge to a local minimum in [19]. Unfortunately this can not be extended to the general RLS-DLA because vector selection is done in an non-optimal way. We will here give a few reflections on the matter leading to some assumptions. Experiments done roughly confirm these assumptions, some are presented in Sec. 3.1.

Let the training vectors be randomly drawn from a set of training vectors, then it follows that the expectation of the norm is constant, $E[\|x_i\|_2] = n_x$. Let the dictionary be normalized, $\|D_i\|_F = \sqrt{K}$ and $\lambda = 1$. In this situation it seems reasonable to assume that $E[\|w_i\|_2] \approx n_w$ and $E[\|r_i\|_2] \approx n_r$, even though w_i is dependent on the vector selection algorithm, thus we lack complete control. The matrix C_i^{-1} in (7) is the sum of positive definite matrices, which implies that $\|C_i^{-1}\|_F$ increases linearly by the iteration number, i, and $\|C_i\|_F$ decreases as $1/i$. From step 5 and 6 in Sec. 2.4 $E[\|u\|_2] \propto 1/i$ and α increases towards

1. Thus, the step size

$$\Delta_i = \|D_i - D_{i-1}\|_F = \|\alpha r u^T\|_F \propto \|r\|_2 / i \tag{23}$$

is decreasing but not fast enough to ensure convergence.

If the steps Δ_i are (mostly) in the same direction the dictionary will drift in that direction. Experiments done, but not shown here, imply that the sum of many steps tends to point roughly in one direction, probably towards a local minimum of the objective function, when the total representation error for the training set is large. The direction is more random when the dictionary is close to a local minimum.

A fixed $\lambda < 1$ makes $\|C_i\|_F$ decrease towards a value larger than zero, and also Δ_i will decrease towards a value larger than zero and thus the algorithm will not strictly converge, unless the representation error is zero.

A simple way to make sure that RLS-DLA converges is to include a factor β in step 7 in Sec. 2.4, i.e. $D_i = D_{i-1} + \alpha\beta r u^T$, and let β (after being 1 for the first iterations) slowly decrease towards zero proportional to $1/i$. Then the step size Δ_i will decrease faster, $\propto 1/i^2$, and the algorithm will converge. However, we do not think this is a good idea, since it probably will not converge to a local minimum. It seems to be better to walk around, with a small Δ_i, and store the best dictionary at all times. The Search-Then-Converge scheme of Sec. 2.3 with an appropriate value for λ_i seems to be a good compromise between searching a sufficiently large part of the solution space and convergence, Fig. 5 and 6.

3 Experiments

This section presents the results of two experiments. First, training vectors from a well defined class of signals, autoregressive (AR) signal, are used. The signal blocks are smoothly distributed in the space $\mathbb{R}^{16}$, without any sparse underlying structure. This experiment is similar to VQ experiments on smoothly distributed data where there is no underlying clustering, the only goal is to minimize the representation error for the training set. Next, in Sec. 3.2 a dictionary is used to make the set of training vectors with a true sparse structure. The goal of the dictionary learning algorithm is to recover this true underlying dictionary from the set of training vectors.

For both the experiments, finite training sets are used, making a fair comparison of the three algorithms possible. The iteration number used in this section is a complete iteration through the whole training set, while in Sec. 2 there was one iteration for each new training vector. The reason for using finite training sets is mainly to be able to compare RLS-DLA to other methods, i.e. ILS-DLA and K-SVD. RLS-DLA is well suited to infinite (very large) training sets. In

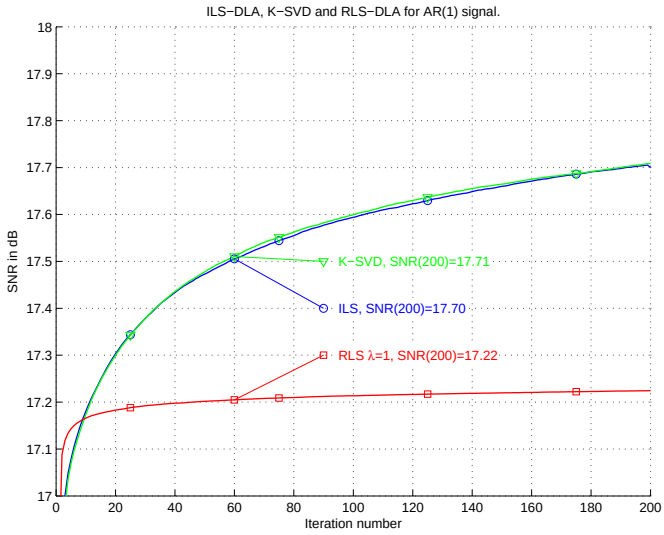

Figure 2: SNR, averaged over 100 trials, is plotted as a function of iteration number for the dictionary learning algorithms ILS-DLA, RLS-DLA and K-SVD.

many cases it is easy to get as many training vectors as wanted, especially for synthetic signals, for example AR(1) signals, but also for many real world signals, like images.

3.1 Sparse representation of an AR(1) signal

In the first experiment we want to explore the algorithm properties in terms of convergence properties, appropriate values for λ_i, and to compare the representation abilities with ILS-DLA and K-SVD. The signal used is an AR(1) signal with no sparse structure. The implementations of the three algorithms are all straight forward, without controlling the frequency of which the individual atoms are used or the closeness of distinct atoms. The K-SVD and the ILS-DLA implementations are identical, except for the few code lines updating the dictionary, see Sec. 1.1. First, the three algorithms are compared. Then, RLS-DLA is further tested using different values for fixed λ and different schemes for increasing λ_i.

For all these tests the same finite training set is used. A set of $L = 4000$ training vectors is made by chopping a long AR(1) signal into vectors of length $N = 16$. The AR(1) signal is generated as $v(k) = 0.95v(k-1) + e(k)$, where $e(k)$ is Gaussian noise with $\sigma = 1$. The number of dictionary atoms is set to $K = 32$, and $s = 4$ atoms are used to approximate each training vector. ORMP is used for vector selection.

We calculate the Signal to Noise Ratio (SNR) during the training process, SNR is $10\log_{10}(\sum_{i=1}^{L}||x_i||^2/\sum_{i=1}^{L}||r_i||^2)$ decibel, where x_i is a training vector and r_i

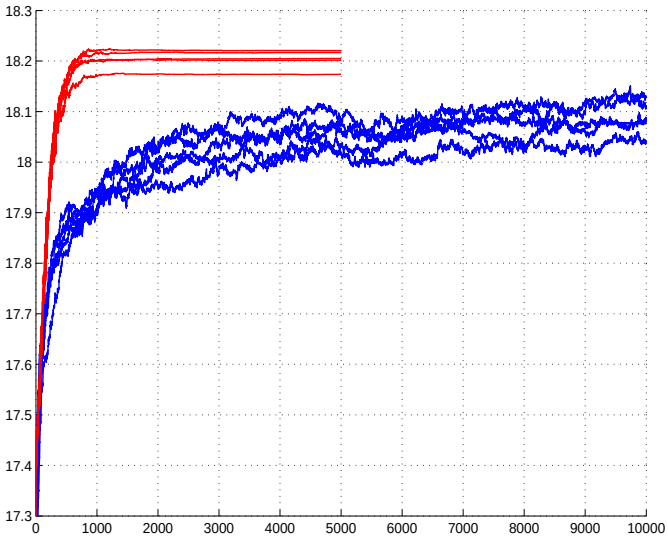

Figure 3: SNR is plotted as a function of iteration number for the dictionary learning algorithm RLS-DLA and ILS-DLA. The 5 plots ending at 5000 iterations are 5 trials of RLS-DLA using adaptive λ_i as in (22), $a = 100$. The 5 plots ending at 10000 iterations are 5 trials of ILS-DLA.

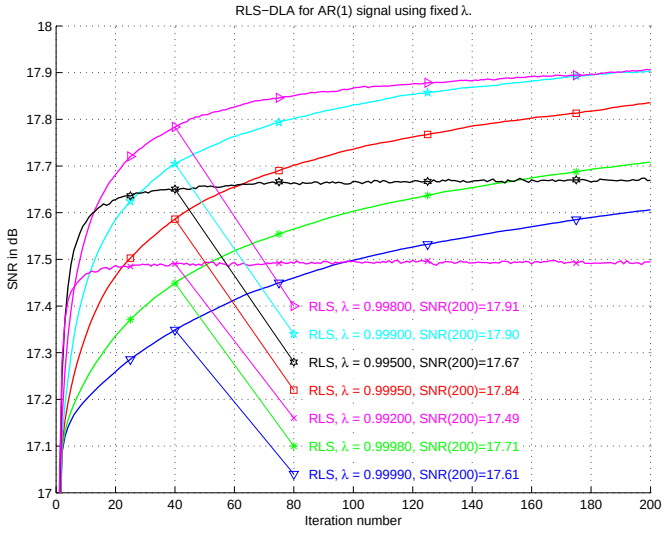

Figure 4: SNR, averaged over 100 trials, is plotted as a function of iteration number for the dictionary learning algorithm RLS-DLA using different fixed values for λ.

is the corresponding representation error using the current dictionary. For ILS-DLA and K-SVD this will be the dictionary of the previous iteration, where one iteration processes the whole training set. For RLS-DLA the dictionary is continuous updated.

Fig. 2 shows how SNR improves during training. 100 trials were done, and the average SNR is plotted. Each trial has a new initial dictionary, selected as

different vectors from the training set, and the order of the training vectors is permuted. The order of the training vectors does not matter for the K-SVD or the ILS-DLA, but it does for the RLS-DLA. These results are not promising for the RLS-DLA, it converges fast and has a better SNR for the first 8 iterations. Then, ILS-DLA and K-SVD continues to improve the SNR while RLS-DLA improves more slowly, at 200 iterations the SNR of K-SVD and ILS-DLA is almost 0.5 decibel better. ILS-DLA and K-SVD performs almost the same, with K-SVD marginally better.

The observant reader has noted that ILS-DLA and K-SVD do not seem to have converged after 200 iterations in Fig. 2. This is indeed true, it can be seen in Fig. 3 where the SNR for thousands of iterations of five individual trials of ILS-DLA and RLS-DLA are shown. ILS-DLA does not converge nicely, the SNR is fluctuating in an erratic way in range between 18 and 18.15. The SNR curves for K-SVD, not shown in Fig. 3, are similar to the ones for ILS-DLA. RLS-DLA, on the other hand, performs much better, it converges quite well, to a higher level after fewer iterations. The adaptive forgetting factor λ_i in the five RLS-DLA trials of Fig. 3 is according to (22) with $a = 100$.

The same 100 trials as used in Fig. 2 are repeated with different fixed values of λ. The results are presented in Fig. 4. Reducing the forgetting factor to $\lambda = 0.9999$ improves SNR after 200 iterations to 17.61 dB. A smaller λ is even better, the algorithm converges faster and to a better SNR, and at $\lambda = 0.9980$ SNR is 17.91. Even smaller λ converge fast but the final SNR values are not as good. Using $\lambda = 0.9998 = 1 - 1/5000$ (which is almost $1 - 1/L$) the SNR curve for RLS-DLA in Fig. 4 is almost identical to the ones for ILS-DLA and K-SVD in Fig. 2. This implies that ILS-DLA and K-SVD "converge" like RLS-DLA with $\lambda \approx 1 - 1/L$.

Finally, the 100 trials are done again with an adaptive λ_i, increasing towards 1 using the different schemes as plotted in Fig. 1. The results are shown in Fig. 5. The adaption strategy seems less important, all schemes producing considerable better results than using different values of fixed λ.

By selecting the parameter a in the adaption schemes for λ_i in (20), (21) or (22), the convergence of the algorithm can be targeted to a given number of iterations. Using the AR(1) training data it is possible to converge to an average SNR at level 18.10 in fewer than 40 iterations and at level 18.17 in fewer than 150 iterations, as can be seen from Fig. 6. From Fig. 3 we see that to reach a level at about 18.20 in SNR requires about 1000 iterations. As seen in Sec. 2.5 the complexity of the three alorithms are comparable.

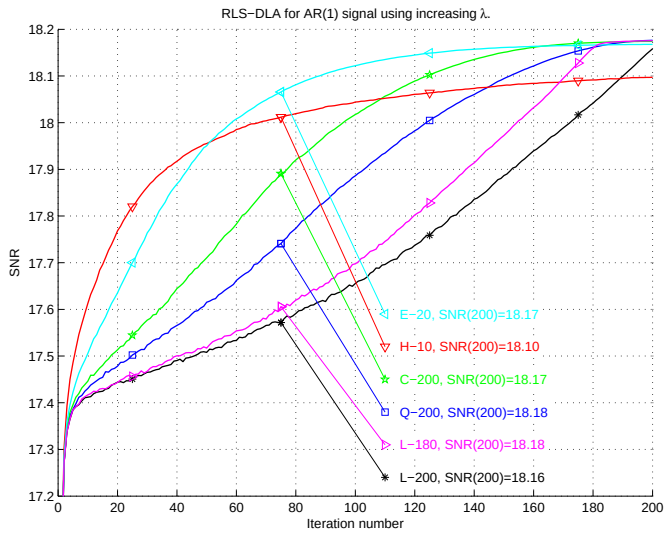

Figure 5: SNR is plotted as a function of iteration number during training with RLS-DLA using the proposed methods for adapting λ_i as shown in Fig. 1. The plot results after averaging over 100 trials.

3.2 Reconstruction of a known dictionary

In the experiment of this section we want to reconstruct a dictionary that produced the set of training vectors. The true dictionary of size $N \times K$, here 20×50, is created by drawing each entry independently from a normal distribution with $\mu = 0$ and $\sigma^2 = 1$, written $\mathcal{N}(0, 1)$. Each column is then normalized to unit norm. The columns are uniformly distributed on the unit sphere in space $\mathbb{R}^N$. A new dictionary is made for each trial and 100 trials are done.

In each trial seven sets of training vectors are made as $X = DW + V$. W is a sparse matrix with $s = 5$ non-zero elements in each column. The values of the non-zero entries are also drawn from $\mathcal{N}(0, 1)$, and their positions in each column are randomly and independently selected. The number of training vectors is 2000 in the first five sets, and then 4000 and 8000 for the last two sets. Finally, Gaussian noise is added, giving signal (DW) to noise (V) ratio (SNR) at 10, 15, 20, 30, 60, 10 and 10 dB for the seven data sets respectively.

Preliminary tests on ILS-DLA, K-SVD, and several variants for RLS-DLA were done. These tests revealed that the performance of ILS-DLA and K-SVD were very similar. Also the different adaptation schemes for λ_i performed very much the same. These preliminary tests showed that 100 iterations through the training set were sufficient for the RLS-DLA, but that ILS-DLA and K-SVD needed at least 200 iterations. The convergence for ILS-DLA and K-SVD is much better here than in the AR(1) experiment. Thus we decided to restrict further testing to ILS-DLA, which is faster than K-SVD, and RLS-DLA with

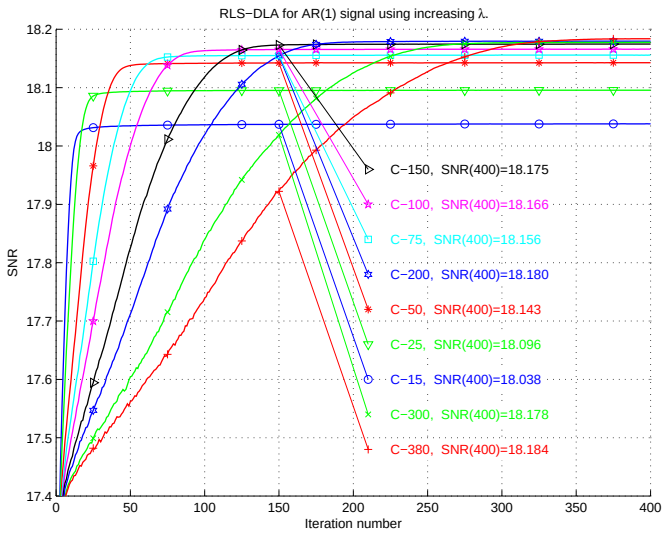

Figure 6: SNR is plotted as a function of iteration number during training with RLS-DLA using the cubic adaptive scheme for λ_i, (20) with $p = 3$. Parameter a, shown as first number in legend, is selected to target the convergence at different number of iterations. The plot results after averaging over 100 trials.

Table 1: Average number of identified atoms, out of 50, using added noise with different SNR, varying from 10 to 60 dB.

		SNR in dB				
Met.	$\cos\beta_{lim}$	10	15	20	30	60
RLS	0.995	8.53	49.61	49.96	49.93	50.00
RLS	0.990	36.69	49.99	49.96	49.93	50.00
RLS	0.980	49.38	49.99	49.96	49.93	50.00
ILS	0.995	4.03	43.91	47.00	47.08	46.89
ILS	0.990	23.17	46.91	47.68	47.53	47.38
ILS	0.980	40.62	47.67	48.07	47.92	47.86

the cubic adaptive scheme for λ_i, (20) with $p = 3$ and $a = 200$. We chose to do 200 iterations in all cases. In the final experiment 1400 dictionaries were learnt, one for each combination of 100 trials, 7 training sets, and 2 methods.

In each case the learnt dictionary $\hat{D}$ is compared to the true dictionary D. The columns, i.e. atoms, are pairwise matched using the Munkres algorithm, also called Hungarian algorithm [33]. Pairwise matching minimizes the sum of angles $\sum_k \beta_k$ (= cost function), where β_k is the angle between atom $d^{(k)}$ in the true dictionary and the matched atom $\hat{d}^{(k)}$ in the learnt dictionary. Since both atoms have 2-norm equal to 1, we have $\cos\beta_k = (d^{(k)})^T\hat{d}^{(k)}$. A positive identification is defined as when this angle β_k is smaller than a given limit β_{lim} and thus the quality of the learnt dictionary may be given as the number of identified atoms (for a given β_{lim}). Pairwise matching by minimizing $\sum_k \beta_k$ does not necessarily maximize the number of identified atoms for a given β_{lim},

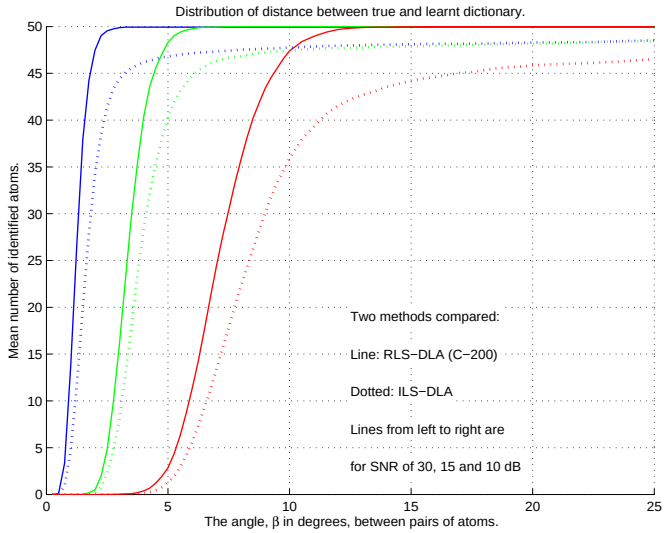

Figure 7: The distance between atoms in true and learnt dictionaries is plotted as a function of β_{lim}.

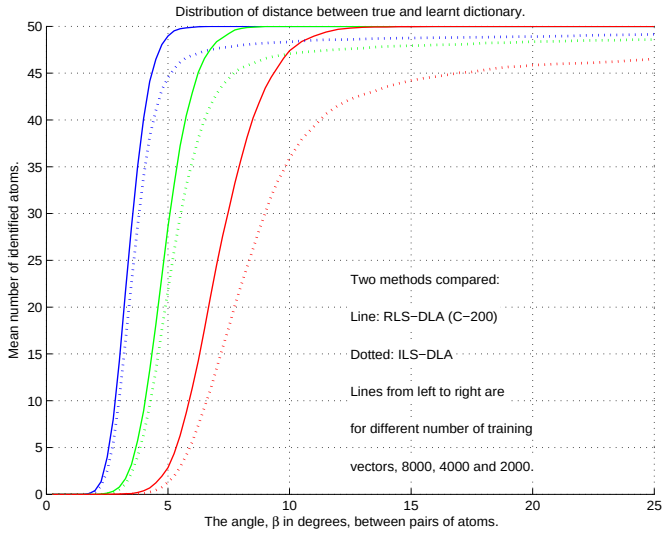

Figure 8: The distance between atoms in true and learnt dictionaries is plotted as a function of β_{lim}.

Table 2: Average number of identified atoms, out of 50, using 2000, 4000 and 8000 training vectors.

		Number of training vectors		
Method	$\cos \beta_{lim}$	2000	4000	8000
RLS	0.995	8.53	40.29	49.84
RLS	0.990	36.69	49.81	50.00
RLS	0.980	49.38	50.00	50.00
ILS	0.995	4.03	32.82	46.64
ILS	0.990	23.17	45.66	48.03
ILS	0.980	40.62	47.42	48.57

but it is a reasonable cost function.

The results, averaged over 100 trials, are presented in Fig. 7, Fig. 8, Table 1 and Table 2. In Fig. 7 and Fig. 8 the distribution of the angles β_k is plotted like the cumulative distribution function (cdf). The y-axis is scaled such that each point on a curve has β_{lim} as x-value and the corresponding number of identified atoms as y-value.

The most important observations are:

- The performance of RLS-DLA is in general considerable better than that of ILS-DLA.
- The result improves as the SNR of the training vectors is increased.
- The result improves as the number of the training vectors is increased.
- The main effect of increasing SNR or the number of training vectors is that the (cdf-like) curves in the figures are moved to the left.
- RLS-DLA identifies all atoms for appropriate values of β_{lim}. This is not the case for ILS-DLA.

The size of the training set is important when searching for the true dictionary. The matrix factorization problem $X = DW$ with W sparse is unique if X has enough columns [34]. It is also often easier to increase the number of training vectors than to reduce noise, especially if the noise contribution is caused by the model, i.e. there is no true underlying dictionary.

Intuitively, as we are familiar only with spaces up to three dimensions, we may think that angles of 20, 10 or even 5 degrees are not a very accurate match. But in $\mathbb{R}^N$ the probability for the angle between two random vectors, uniform on the unit sphere, has a pdf proportional to $\sin^{N-2} \beta$, which give the following values for the cdf when $N = 20$, $F(10) = 6.18 \cdot 10^{-16}$, $F(15) = 1.31 \cdot 10^{-12}$, $F(30) = 3.92 \cdot 10^{-7}$, and finally $F(45) = 3.38 \cdot 10^{-4}$.

4 Conclusion

We have presented a new algorithm for dictionary learning, RLS-DLA. It is closely related both to k-means and RLS, and to earlier presented dictionary learning algorithms ILS-DLA and K-SVD. The dictionary is updated continuously as each new training vector is processed. The difference, measured by the Frobenius norm, between two consecutive dictionaries in the iterations goes towards zero as the iteration number increases, but not fast enough to ensure strict convergence. Nevertheless, supported by the experiments, we conclude that the RLS-DLA is a sound algorithm with good convergence properties.

An adaptive forgetting factor λ_i can be included making the algorithm flexible. λ_i can be tuned to achieve good signal representation in few iterations, or to achieve an even better representation using more iterations, as seen in Fig. 6. An important advantage of RLS-DLA is that the dependence of the initial dictionary can be reduced simply by gradually forgetting it. The experiments performed demonstrate that RLS-DLA is superior to ILS-DLA and K-SVD both in representation ability of the training set and in the ability of reconstruction of a true underlying dictionary. It is easy to implement, and the running time is between that of ILS-DLA and K-SVD. Another advantage of RLS-DLA is the ability to use very large training sets, this leads to a dictionary that can be expected to be general for the used signal class, and not a dictionary specialized to the particular (small) training set used.

There is still research to be done on this new algorithm. A more complete convergence analysis would be helpful to better understand the algorithm and to better judge if it is appropriate for a given problem. Further experiments, especially on real world signals like images, are needed to gain experience and to make a more complete comparison of RLS-DLA with ILS-DLA and K-SVD as well as other algorithms. RLS-DLA, having $\lambda < 1$, is (like RLS) an *adaptive* algorithm. Future work may identify applications where this adaptability is useful.

References

[1] K. Engan, B. Rao, and K. Kreutz-Delgado, "Frame design using FOCUSS with method of optimized directions (MOD)," in *Proc. NORSIG '99*, Oslo, Norway, Sep. 1999, pp. 65–69.

[2] S. F. Cotter and B. D. Rao, "Application of total least squares (TLS) to the design of sparse signal representation dictionaries," in *Proc. Asilomar Conf. on Signals, Systems and Computers*, vol. 1, Monterey, California, Nov. 2002, pp. 963–966.

[3] K. Kreutz-Delgado, J. F. Murray, B. D. Rao, K. Engan, T.-W. Lee, and T. J. Sejnowski, "Dictionary learning algorithms for sparse representation," *Neural Comput.*, vol. 15, no. 2, pp. 349–396, Feb. 2003.

[4] 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. [Online]. Available: http://dx.doi.org/10.1109/TSP.2006.881199

[5] 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.

[6] B. Mailhé, S. Lesage, R. Gribonval, and F. Bimbot, "Shift-invariant dictionary learning for sparse representations: Extending K-SVD," in *Proceedings of the 16th European Signal Processing Conference (EUSIPCO-2008)*, Lausanne, Switzerland, aug 2008.

[7] M. Yaghoobi, T. Blumensath, and M. E. Davies, "Dictionary learning for sparse approximations with the majorization method," *IEEE Trans. Signal Processing*, vol. 109, no. 6, pp. 1–14, Jun. 2009.

[8] 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: ACM, jun 2009, pp. 689–696.

[9] K. Engan, S. O. Aase, and J. H. Husøy, "Designing frames for matching pursuit algorithms," in *Proc. ICASSP '98*, Seattle, USA, May 1998, pp. 1817–1820.

[10] ——, "Method of optimal directions for frame design," in *Proc. ICASSP '99*, Phoenix, USA, Mar. 1999, pp. 2443–2446.

[11] K. Skretting, "Sparse signal representation using overlapping frames," Ph.D. dissertation, NTNU Trondheim and Høgskolen i Stavanger, Oct. 2002, available at http://www.ux.uis.no/ karlsk/.

[12] K. Skretting, J. H. Husøy, and S. O. Aase, "General design algorithm for sparse frame expansions," *Elsevier Signal Processing*, vol. 86, pp. 117–126, Jan. 2006.

[13] ——, "A simple design of sparse signal representations using overlapping frames," in *Proc. 2nd Int. Symp. on Image and Signal Processing and Analysis, ISPA01*, Pula, Croatia, Jun. 2001, pp. 424–428, available at http://www.ux.uis.no/ karlsk/.

[14] K. Skretting, K. Engan, J. H. Husøy, and S. O. Aase, "Sparse representation of images using overlapping frames," in *Proc. 12th Scandinavian Conference on Image Analysis, SCIA 2001*, Bergen, Norway, Jun. 2001, pp. 613–620, also available at http://www.ux.uis.no/ karlsk/.

[15] D. R. Cox, "Note on grouping," *J. Amer. Statist. Assoc.*, vol. 52, pp. 543–547, 1957.

[16] S. P. Lloyd, "Least squares quantization in PCM," pp. 129–137, 1957, published in March 1982, IEEE Trans. Inf. Theory.

[17] E. Forgy, "Cluster analysis of multivariate data: Efficiency vs. interpretability of classification," *Biometrics*, vol. 21, p. 768, 1965, abstract.

[18] Y. Linde, A. Buzo, and R. M. Gray, "An algorithm for vector quantizer design," *IEEE Trans. Commun.*, vol. 28, pp. 84–95, Jan. 1980.

[19] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, vol. I, 1967, pp. 281–296.

[20] G. Davis, "Adaptive nonlinear approximations," Ph.D. dissertation, New York University, Sep. 1994.

[21] 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.

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

[23] 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.

[24] Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad, "Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition," in *Proc. of Asilomar Conference on Signals Systems and Computers*, Nov. 1993.

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

[26] I. F. Gorodnitsky and B. D. Rao, "Sparse signal reconstruction from limited data using FOCUSS: A re-weighted minimum norm algorithm," *IEEE Trans. Signal Processing*, vol. 45, no. 3, pp. 600–616, Mar. 1997.

[27] K. Engan, B. Rao, and K. Kreutz-Delgado, "Regularized FOCUSS for subset selection in noise," in *Proc. of NORSIG 2000*, Sweden, Jun. 2000, pp. 247–250.

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

[29] Z. Xie and J. Feng, "KFCE: A dictionary generation algorithm for sparse representation," *Elsevier Signal Processing*, vol. 89, pp. 2072–2079, Apr. 2009.

[30] C. Darken and J. E. Moody, "Note on learning rate schedules for stochastic optimization," in *NIPS*, 1990, pp. 832–838.

[31] M. Y. Mashor, "Improving the performance of k-means clustering algorithm to position the centres of RBF network," *International Journal of the Computer, the Internet and Management*, pp. 1–15, Aug. 1998.

[32] L. Bottou and Y. Bengio, "Convergence properties of the K-means algorithms," in *Advances in Neural Information Processing Systems*, G. Tesauro, D. Touretzky, and T. Leen, Eds., vol. 7. The MIT Press, 1995, pp. 585–592. [Online]. Available: citeseer.nj.nec.com/bottou95convergence.html

[33] J. Munkres, "Algorithms for the assignment and transportation problems," *Journal of the Society for Industrial and Applied Mathematics*, vol. 5, no. 1, pp. 32–38, Mar. 1957.

[34] 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.