

SMALLbox - An Evaluation Framework for Sparse Representations and Dictionary Learning Algorithms

Ivan Damnjanovic¹, Matthew E.P. Davies¹, Mark D. Plumbley¹

¹ Queen Mary University of London, Centre for Digital Music,
Mile End Road, London, E1 4NS, United Kingdom
{ivan.damnjanovic, matthew.davies, mark.plumbley}@elec.qmul.ac.uk

Abstract. SMALLbox is a new foundational framework for processing signals, using adaptive sparse structured representations. The main aim of SMALLbox is to become a test ground for exploration of new provably good methods to obtain inherently data-driven sparse models, able to cope with large-scale and complicated data. The toolbox provides an easy way to evaluate these methods against state-of-the-art alternatives in a variety of standard signal processing problems. This is achieved through a unifying interface that enables a seamless connection between the three types of modules: problems, dictionary learning algorithms and sparse solvers. In addition, it provides interoperability between existing state-of-the-art toolboxes. As an open source MATLAB toolbox, it can be also seen as a tool for reproducible research in the sparse representations research community.

Keywords: Sparse representations, Dictionary learning, Evaluation framework, MATLAB toolbox

1 Introduction

Sparse representations has become a very active research area in recent years and many toolboxes implementing a variety of greedy or other types of sparse algorithms have become freely available in the community [1-4]. As the number of algorithms has grown, there has become a need for a proper testing and benchmarking environment. This need was partially addressed with the SPARCO framework [5], which provides a large collection of imaging, signal processing, compressed sensing, and geophysics sparse reconstruction problems. It also includes a large library of operators that can be used to create new test problems. However, using SPARCO with other sparse representations toolboxes, such as SparseLab [1] is non-trivial because of inconsistencies in the APIs of the toolboxes.

Many algorithms exist that aim to solve the sparse representation dictionary learning problem [6-7, 10]. However, no comprehensive means of testing and benchmarking these algorithms exists, in contrast to the sparse representation problem when the dictionary is known. The main driving force for this work is the lack of a toolbox similar to SPARCO for dictionary learning problems. Recognising the need of the community for such a toolbox, we set out to design a MATLAB toolbox with three main aims:

- to enable an easy way of comparing dictionary learning algorithms,
- to provide a unifying API that will enable interoperability and re-use of already available toolboxes for sparse representation and dictionary learning,
- to aid the reproducible research effort in the sparse signal representations and dictionary learning [12].

In the section 2 of this paper, we give a short overview of sparse representations and dictionary learning. In section 3, the SMALLbox toolbox design approach is presented, with implementation details in section 4, followed by usage examples in section 5 and conclusions in section 6.

2 Sparse Representations and Dictionary Learning

One of the main requirements in many signal processing applications is to represent the signal in a transformed domain where it can be expressed as a linear combination of a small number of coefficients. In many research areas such as compressed sensing, image de-noising and source separation, these sparse structured signal representations are sought-after signal models. Depending on the application, we seek either an exact solution for a noise-free models or an approximate sparse reconstruction of the signal in the presence of noise:

$$\mathbf{b} = \mathbf{A}\mathbf{x} \quad (1)$$

or

$$\mathbf{b} = \mathbf{A}\mathbf{x} + \mathbf{n} \quad (2)$$

where $\mathbf{b} \in R^m$ is the signal of interest, $\mathbf{A} \in R^{m \times n}$ is transformation matrix (or dictionary), $\mathbf{x} \in R^n$ is the sparse coefficients vector and $\mathbf{n} \in R^m$ is noise vector. When $m < n$ the problem is *undetermined* and there is no unique solution. Evidently, additional constraint needs to be imposed to the signal model to find the solution of interest. In this sense, probably the most studied and well-understood constraint is to assume a Gaussian distribution on the coefficients and to minimise the l_2 norm of the vector \mathbf{x} . However, in applications such as compression for example, it is more appropriate to impose the sparsity assumption on the coefficients, i.e. to minimise the l_0 norm of the vector \mathbf{x} . Since l_0 norm minimisation is known to be an NP-hard problem, one can try finding an approximate solution using greedy algorithms such as Matching Pursuit (MP) [8] or Orthogonal Matching Pursuit (OMP) [9]. Alternatively, we can relax the sparsity assumption by imposing a Laplace distribution on the coefficients and minimise the l_1 norm, which can be solved using different convex optimisation methods.

In the SPARCO toolbox, the problems to be solved are given through a consistent interface represented in the form of a problem structure that contains a measurement vector \mathbf{b} , operator \mathbf{A} and all other components of the test problem. Operator \mathbf{A} is given in the following form:

$$\mathbf{A} = \mathbf{M}\mathbf{B} \quad (3)$$

The measurement operator \mathbf{M} describes how the signal was sampled and operator \mathbf{B} represents a basis with which the signal can be sparsely represented [5]. It is assumed that the basis that can give a sparse solution is known in advance. The success of the sparse representation heavily depends on the choice of the basis and the transform dictionary \mathbf{A} and how well the dictionary reflects the structure to be found in the signal. Learning the matrix \mathbf{A} from the data itself is a key to finding a sparse representation of the new classes of data. Dictionary learning for a sparse representation can be formulated as a problem of the following type:

$$\min_{\mathbf{A}, \mathbf{x}} \|\mathbf{Y} - \mathbf{A}\mathbf{X}\|_F^2 \quad \text{subject to} \quad \forall i \quad \|\mathbf{x}_i\|_0 \leq s \quad (4)$$

where \mathbf{Y} is a matrix with vectors of training data and \mathbf{x}_i are sparse representations of the training vectors. We want to choose a transform matrix \mathbf{A} that will minimise the residual, given that the training data representation vectors \mathbf{x}_i are sparse with a maximum of s non-zero coefficients.

Reflecting high activity in the research area, many dictionary learning algorithms are available, but currently no evaluation framework exists for testing them.

3 Design approach to SMALLbox

The SMALLbox framework is designed to fulfil two main goals: (1) to provide a set of test problems that permit formative evaluation of the techniques and algorithms to be developed elsewhere, and (2) to be a framework within which to build demonstrator applications. The design of the SMALLbox toolbox was constructed to allow easy portability of existing algorithms and new algorithms to be developed, taking into account the experiences in using toolboxes such as SPARCO [5] and SparseLab [1]. A graphical overview of the design of SMALLbox is shown in Fig 1.

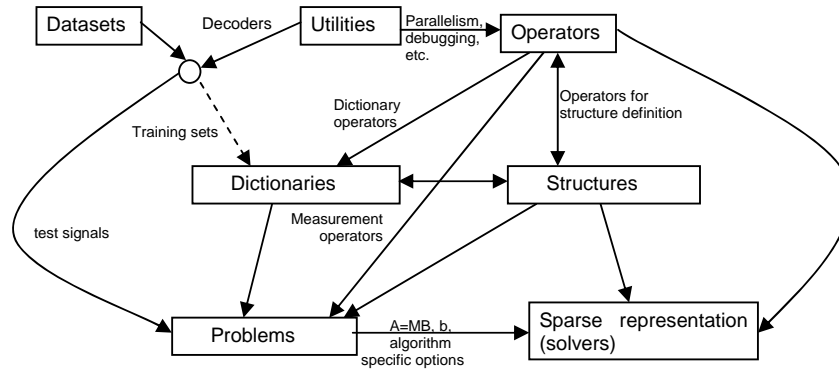


Fig. 1. Design of the Evaluation Framework.

The main interoperability of the design is given through the *Problems* part that can be defined either as sparse representations or dictionary learning. In generating a problem, some *utilities* can be used to decode a *dataset* and prepare a *test signal* or a

training set for dictionary learning. The *dictionaries* can be either defined or learned using dictionary learning algorithms. In the former case, they can be given as implicit dictionaries as a combination of the given *operators* and *structures* or explicit in the form of dictionary matrix. In the latter case, they are learned from the training data. Once the *dictionary* is set in the *problem*, the *problem* is ready to be solved by one of the *sparse representation* algorithms.

SMALLbox is designed to enable an easy exchange of information and comparison of different modules developed through a unified API data structure. The structure is made to fulfil two main goals. The first goal is to separate typical sparse signal processing problems into three meaningful units:

- a) problem specification (preparing data for learning the structures, representation and reconstruction),
- b) dictionary learning (using a prepared training set to learn the natural structures in the data) and
- c) sparse representation (representing the signal with a pre-specified or learned dictionary).

The second goal is a seamless connection between the three types of modules and ease of communication of data between the problem, dictionary learning and sparse representation parts of the structure.

4 SMALLbox Implementation

The evaluation framework is implemented as a Matlab toolbox called SMALLbox. To enable easy comparison with the existing state-of-the-art algorithms, during the installation procedure SMALLbox checks the Matlab path for existence of the following freely available toolboxes and will automatically download and install them, as required:

- SPARCO (v.1.2) - set of sparse representation problems[5]
- SparseLab (v.2.1) - set of sparse solvers [1]
- Sparsify (v.0.4) - set of greedy and hard thresholding algorithms [2]
- SPGL1 (v.1.7) - large-scale sparse reconstruction solver [3]
- GPSR (v.6.0) - Gradient projection for sparse reconstruction [4]
- KSVD-box (v.13) and OMP-box (v.10) - dictionary learning [6]
- KSVDS-box (v.11) and OMPS-box (v.1) - sparse dictionary learning [7].¹
- SPAMS - Online dictionary learning [10]²

The SMALLbox provides a “glue” structure to allow algorithms from those toolboxes to be used with a common API. The structure consists of three main sub-structures: **Problem** structure, **DL** (dictionary learning) structures and **solver** structures. Since the **Problem** structure is designed to be backward compatible with the SPARCO problem structure [5], it can be filled with SPARCO generateProblem or one of the dictionary learning problems provided in SMALLbox. If the problem is dictionary learning, one or more **DL** structures can be specified, so [6-7] or any other

¹ The list of 3rd party toolboxes included in SMALLbox version 1.0

² An API for SPAMS is included, but due to licensing issues this toolbox needs to be installed by the user.

dictionary learning technique can be compared with specified set of parameters. Finally, to sparsely represent the signal in a dictionary (either defined in the **Problem** structure or learned in the previous step), one or more **solver** structures can be used to specify any solver from [1-4] or any of the solvers provided in SMALLbox.

4.1 Generating Problems (**Problem** structure)

The **Problem** structure defines all necessary aspects of a problem to be solved. To be compatible with the SPARCO, it needs to have five fields defined prior to any sparse representation of the data:

- A – a matrix or operator representing dictionary in which signal is sparse
- b – a vector or matrix representing signal or signals to be represented
- *reconstruct* – a function handle to reconstruct the signal from coefficients
- *signalSize* – the dimension of the signal
- *sizeA* – if matrix A is given as an operator the size of the dictionary needs to be defined in advance.

Other fields that further describe the problem, which are useful for either reconstruction of the signal or representation of the results, might be generated by the SPARCO `generateProblem` function or the SMALLbox problem functions. The new problems implemented in the SMALLbox version 1.0 are: Image De-noising [6], Automatic Music Transcription [11] and Image Representation using another image as a dictionary.

In the case of a dictionary learning problem, fields A and *reconstruct* are not defined while generating the problem, but after the dictionary is learned and prior to the sparse representation. In this case, field b needs to be given in matrix form to represent the training data and another field p defining the number of dictionary elements to be learned needs to be specified.

4.2 Dictionary Learning (**DL** structure)

The structure for dictionary learning - **DL** is a structure that defines dictionary learning algorithm to be used. It is initialised with a utility function `SMALL_init_DL`, which will define five mandatory fields:

- *toolbox* - a field used to discriminate the API
- *name* - the name of dictionary learning function from the particular toolbox
- *param* - a field containing parameters for the particular DL technique and in the form given by the toolbox API
- D - a field where the learned dictionary will be stored
- *time* - a field to store learning time.

After *toolbox*, *name* and *param* fields are set, the function `SMALL_learn` is called with **Problem** and **DL** structures as inputs. According to the `DL.toolbox` field, the function calls the `DL.name` algorithm with its API and outputs learned dictionary D and *time* spent. The `DL.param` field contains parameters such as dictionary size, the number of iterations, the error goal or similar depending on the particular algorithm

used. To compare a new dictionary learning algorithm against existing ones, the algorithm needs to be in the MATLAB path and introduced to SMALLbox by defining two parameters *<Toolbox ID>* and *<Preferred API>* in the `SMALL_learn` function, where examples and a simple explanation are provided. Once the new dictionary is learned, field *A* of the *Problem* structure is defined to be equal to *DL.D* and also the reconstruction function is instructed to use this particular dictionary. In this way, a SPARCO compatible Problem structure is defined and ready for use.

4.3 Sparse Representation (*solver* structure)

Similar to dictionary learning every instance of the sparse representation needs to be initialised with the `SMALL_init_solver` function. It will define mandatory fields of the *solver* structure:

- *toolbox* - a field with toolbox name (e.g. sparselab)
- *name* - the name of solver from the particular toolbox (e.g. SolveOMP)
- *param* - the parameters in the form given by the toolbox API
- *solution* - the output representation
- *reconstructed* - the signal reconstructed from solution
- *time* - the time spent for sparse representation.

With the input parameters of the *solver* structure set, the `SMALL_solve` function is called with *Problem* and *solver* structure as inputs. The function calls *solver.name* algorithm with API specified by *solver.toolbox* and outputs *solution*, *reconstructed* and *time* fields. To introduce a new sparse representation algorithm it needs to be in the MATLAB path and *<Toolbox ID>* and *<Preferred API>* need to be defined for the algorithm in the `SMALL_solve` function. Three solvers that can find a sparse representation of the whole training set matrix in one go are included in SMALLbox (`SMALL_MP`, `SMALL_chol` and `SMALL_cgp`).

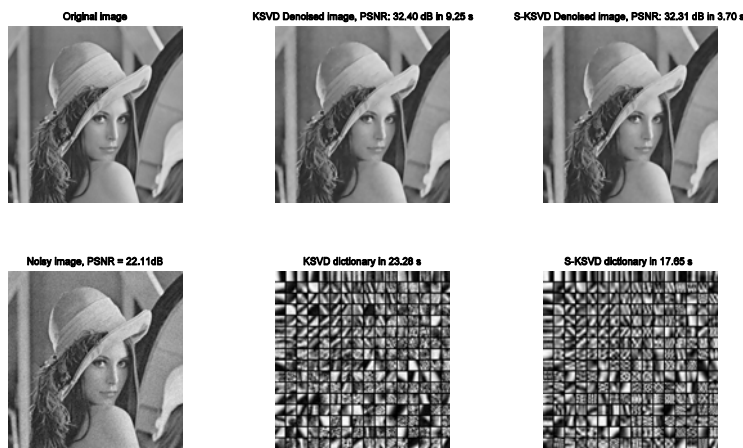


Fig. 2. SMALLbox example results - K-SVD [6] versus S-KSVD [7] in image de-noising.

5 Examples

As a part of SMALLbox, a variety of examples on how included dictionary learning and sparse representation techniques can be used and compared on SPARCO and SMALLbox problems. As an example, `small_solver_test.m` will generate SPARCO problem 6 (sparse representation of \mathbf{b} – a piecewise cubic polynomial signal, in \mathbf{B} – a Daubechies basis with \mathbf{M} – a Gaussian ensemble measurement matrix), test four solvers on the problem (SMALL_cgp, SMALL_chol, Solve_OMP from SparseLab and `greed_pcg` from Sparsify), show computational time and plot solutions and reconstructed signals against the original.

Two examples of dictionary learning for image de-noising are presented in Figs 2 and 3. In the first example, we compared the KSVD algorithm [6] with S-KSVD [7]. The main idea presented in [7] is that if an implicit dictionary (in this case an overcomplete DCT) is used as base dictionary on which the sparse dictionary is learned, much better computational time can be achieved while still keeping adaptability and performance characteristics of explicit dictionaries. The example and results in Figure 2 support this claim. De-noising in the S-KSVD is almost 3 times faster while the PSNR is only 0.09 dB lower.

The example in Figure 3 presents a comparison of online dictionary learning [10] and KSVD [6] and shows how SMALLbox can be used to easy change the parameters of the problem (in this case the training size). It supports the claim from [10] that, in contrast to iterative algorithms [6], online dictionary learning does not depend on the size of the training set.

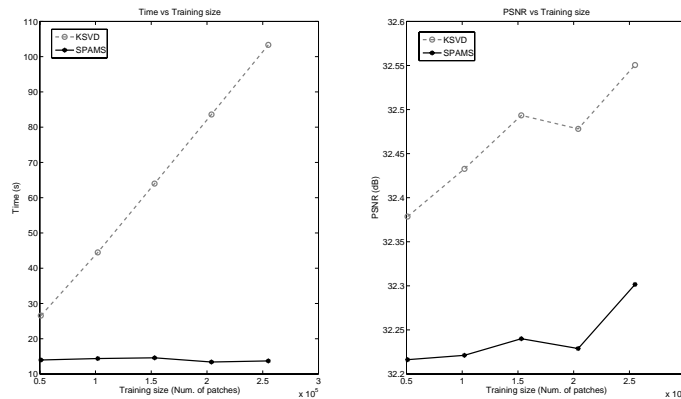


Fig. 3. SMALLbox example results - K-SVD [6] versus SPAMS [10] with variable training size.

6 Conclusions

We have introduced SMALLbox - an Evaluation Framework that enables easy prototyping, testing and benchmarking of sparse representation and dictionary

learning algorithms. This is achieved through a set of test problems and an easy evaluation against state-of-the-art algorithms. As a part of the EU FET SMALL project, more problems, solvers and dictionary learning techniques that are developed will be included in SMALLbox as the project proceeds.

For instructions how to download the SMALLbox and reproduce the figures in this paper, please visit: <http://small-project.eu/>.

Acknowledgments. This research is supported by EU FET-Open ProjectFP7-ICT-225913 “SMALL”, and ESPRC Platform Grant EP/045235/1.

The authors would also like to thank to all researchers working on the SMALL project especially Miki Elad and Pierre Vandergheynst for fruitful discussion and help in developing the SMALLbox.

References

1. Donoho, D., Stodden, V., Tsaig, Y.: Sparselab. 2007, <http://sparselab.stanford.edu/>
2. Blumensath, T., Davies, M. E.: Gradient pursuits. In *IEEE Transactions on Signal Processing*, vol. 56, no. 6, pp. 2370–2382, June 2008.
3. Berg, E. v., Friedlander, M. P.: Probing the Pareto frontier for basis pursuit solutions. In *SIAM Journal on Scientific Computing*, vol. 31, no. 2, pp. 890–912, 2008.
4. Figueiredo, M. A. T., Nowak, R. D., Wright, S. J.: Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems. In *Journal of Selected Topics in Signal Processing: Special Issue on Convex Optimization for Signal Processing*, December 2007.
5. Berg, E. v., Friedlander, M. P., Hennenfent, G., Herrmann, F., Saab, R., Yilmaz, O.: SPARCO: A testing framework for sparse reconstruction. In *ACM Trans. on Mathematical Software*, 35(4):1-16, February 2009.
6. Aharon, M., Elad, M., Bruckstein, A. M.: The K-SVD: An algorithm for designing of overcomplete dictionaries for sparse representation. In *IEEE Transactions on Signal Processing*, vol. 54, no. 11, pp. 4311–4322, 2006.
7. Rubinstein, R., Zibulevsky, M. and Elad, M.: Double Sparsity: Learning Sparse Dictionaries for Sparse Signal Approximation. In *IEEE Transactions on Signal Processing*, Vol. 58, No. 3, Pages 1553-1564, March 2010.
8. Mallat, S. G., Zhang, Z.: Matching Pursuits with Time-Frequency Dictionaries. In: *IEEE Transactions on Signal Processing*, December 1993, pp. 3397-3415
9. Pati, Y. C., Rezaiifar, R., Krishnaprasad, P. S.: Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. In *Proc. 27th Asilomar Conference on Signals, Systems and Computers*, A. Singh, ed., Los Alamitos, CA, USA, IEEE Computer Society Press, (1993)
10. Mairal, J., Bach, F., Ponce J., Sapiro, G.: Online Learning for Matrix Factorization and Sparse Coding. In *Journal of Machine Learning Research*, volume 11, pages 19-60. 2010.
11. Bertin, N., Badeau, R., Richard, G.: Blind signal decompositions for automatic transcription of polyphonic music: NMF and K-SVD on the benchmark. In *Proc. Of International Conference on Acoustics, Speech, and Signal Processing ICASSP'07*, Honolulu, Hawaii, USA, April 15-20, 2007, vol. I, pp. 65-68
12. Vandewalle, P., Kovacevic, J., Vetterli, M.: Reproducible Research in Signal Processing - What, why, and how. In *IEEE Signal Processing Magazine*, 26 (3). pp. 37-47., (2009).