Rw-weighted ℓ_1 Spatial Filtering Library for MATLAB

Adam S. Charles

May 14, 2015

1 Introduction

Sparsity-based signal decompositions have recently been used in many applications from inverse problems in image processing (i.e. de-blurring, de-noising or inpainting) [6] to novel imaging procedures such as compressive sensing [2]. These methods typically rely on decomposing a signal $\boldsymbol{x}$ into a small number of atoms from a potentially over-complete dictionary of atoms $\boldsymbol{\Psi}$. Concisely we can write

$$\boldsymbol{x} = \boldsymbol{\Psi}\boldsymbol{a} + \boldsymbol{\epsilon},$$

where $\boldsymbol{a}$ is the coefficient vector which is assumed to be mostly zeros and $\boldsymbol{\epsilon}$ is the modeling error. We can recover this sparse coefficient vector from the signal $\boldsymbol{x}$ via the basis-pursuit de-noising (BPDN) optimization problem

$$\widehat{\boldsymbol{a}} = \arg\min_{\boldsymbol{a}} \left\|\boldsymbol{x} - \boldsymbol{\Psi}\boldsymbol{a}\right\|_2^2 + \gamma \left\|\boldsymbol{a}\right\|_1.$$

In the case where $\boldsymbol{x}$ is observed indirectly via a linear measurement operation $\boldsymbol{\Phi}$ as

$$\boldsymbol{y} = \boldsymbol{\Phi}\boldsymbol{x} + \boldsymbol{\epsilon} = \boldsymbol{\Phi}\boldsymbol{\Psi}\boldsymbol{a} + \boldsymbol{\epsilon},$$

where $\boldsymbol{\epsilon}$ is now a combination of potential modeling and measurement errors, a similar optimization program recovers the sparse coefficients:

$$\widehat{\boldsymbol{a}} = \arg\min_{\boldsymbol{a}} \left\|\boldsymbol{x} - \boldsymbol{\Psi}\boldsymbol{a}\right\|_2^2 + \gamma \left\|\boldsymbol{a}\right\|_1.$$

One alternative to BPDN uses an iterative procedure to obtain more accurate solutions. While lacking the precise recovery guarantees available for BPDN (see [1]), re-weighted ℓ_1 (RWL1) has been observed to empirically recover sparse coefficients more accurately than BPDN [3, 8]. RWL1 works by weighting coefficients in a BPDN-style optimization procedure based on confidence in the coefficient being active and large. Specifically, RWL1 alternates between solving the weighted BPDN

$$\widehat{\boldsymbol{a}}^t = \arg\min_{\boldsymbol{a}} \left\|\boldsymbol{x} - \boldsymbol{\Psi}\boldsymbol{a}\right\|_2^2 + \lambda_0 \left\|\boldsymbol{\Lambda}^{t-1}\boldsymbol{a}\right\|_1.$$

$$\boldsymbol{\lambda}[k]^t = \frac{\alpha}{|\widehat{\boldsymbol{a}}^t[k]| + \beta}$$

where λ_0, α and β are pre-set parameter, t is the algorithmic iteration, and $\boldsymbol{\Lambda} = \text{diag}(\boldsymbol{\lambda})$ is a diagonal matrix consisting of the weights trading off the different coefficients. Larger weights enforce stricter sparsity, encouraging coefficients to be zero, while lower weights encourage coefficients to be non-zero in the signal description. The re-weighting step amplifies this effect at each iteration, further encouraging active coefficients to have less penalized values and in-active coefficients to be more heavily penalized.

1.1 Re-weighted ℓ_1 with Spatial Filtering

While these algorithms have worked well for single-signal inference, many datasets consist of many, correlated signals. As a special case, this code package considers the two-dimensional case where spatially correlated signals are to undergo inference all at once. Mathematically, we can index each spatial location as $\{i, j\}$, and that each spatially correlated vector can be considered independent *given the underlying sparse coefficients*:

$$\boldsymbol{x}_{i,j} = \sum \boldsymbol{\phi}_k \boldsymbol{a}_{i,j}[k] + \boldsymbol{\epsilon}_{i,j} = \boldsymbol{\Phi} \boldsymbol{a}_{i,j} + \boldsymbol{\epsilon}_{i,j},$$

Likewise, in this code package, we also consider the measurements as spatially independent when conditioned on the sparse coefficients:

$$\boldsymbol{y}_{i,j} = \boldsymbol{\Psi} \boldsymbol{x}_{i,j} = \boldsymbol{B} \boldsymbol{\Phi} \boldsymbol{a}_{i,j} + \widetilde{\boldsymbol{\epsilon}}_{i,j},$$

While in general, more complex measurement systems can be considered, i.e.

$$\boldsymbol{y}_{i,j} = \psi(\{\boldsymbol{x}_{i,j}\}) \tag{1}$$

where $\psi(\cdot)$ is potentially a function of *all* $\boldsymbol{x}$ at *all* locations, here we consider the independent case. This case is still relevant in many applications and allows us to leverage parallelization in the implementation. Although each measurement vector can be used to independently to recover the coefficients at each spatial location via the BPDN optimization, greater accuracy can be achieved by leveraging spatial dependencies between the sparse coefficients. One such dependence is that the coefficient values might have spatial consistency, i.e. the values of the k^{th} coefficient over the spatial indices $\{i, j\}$ ($\boldsymbol{a}_{i,j}[k]$) may change slowly, or tend to be clumped together. This package leverages these dependencies via the RWL1 optimization to create a RWL1 spatial filtering (RWL1-SF) algorithm, as presented in [5]. Essentially, the algorithm function much as the RWL1 algorithm would, if run independently at each location), with the one change that the update for the regularization parameter at each index and spatial location include an extra term in the denominator as:

$$\boldsymbol{\lambda}_{i,j}[k] = \frac{\alpha}{\left|[\boldsymbol{K} * \boldsymbol{A}_k]_{i,j}\right| + \beta}$$

where the term $[\boldsymbol{K} * \boldsymbol{A}_k]_{i,j}$ represents the $\{i, j\}^{th}$ term of the kernel $\boldsymbol{K} \in \mathbb{R}^{L \times P}$ convolved with the spatial field of previous estimates for the k^{th} coefficient. Note that while this spatial regularization can accumulate weak evidence spread over several neighboring pixels to perform inference, the model does not force spatial homogeneity so that single-pixel (or sub-pixel) objects are missed. In other words, rather than low-pass filtering the estimates of interest (the $a_{i,j,k}$ variables), the spatial averaging is applied to a second order variable ($\gamma_{i,j,k}$) that simply biases a sparse inference process. In fact, though an explicit test with single-pixel anomalies is beyond the scope of this letter, previous work using this approach for dynamic filtering [4] showed that this method of stochastic filtering is particularly robust to model mismatch.

Table 1: Functions included in the Dctionary Learning Library

Function Name	Description
`gen_multi_infer`	Run BPDN for multiple vectors (in parallel)
`gen_rw_multi_infer`	Run RWL1 for multiple vectors (in parallel)
`rwl1sf_infer`	Run RWL1-SF for a data cube
`HSI_RWinfer_test`	Test script to run `HSI_RWinfer`
`l1ls_nneg_wrapper`	Wrapper for l1ls with non-negativity constraints
`HS_image`	Function that reshapes HSI vectors in a 3D array
`recom_best`	Permutes a matrix's rows and columns to maximize the diagonal sum
`sangle`	Function to evaluate spectral angles
`HSI2MSI`	Converts an HSI image to a simulated MSI image
`match_vecs`	Function that matches one set of vectors with a reference set
`compare_ix`	Counts the value matches between two vectors

2 Code Functionality

The code in this library is aimed at allowing a user to input a dataset and to extract the spatially varying sparse coefficients via the RWL1-SF optimization procedure [5]. The BPDN sub-problem is solved using functions included in the `l1_ls` package [7]. Other methods can be used instead by changing the `l1ls_nneg_wrapper` function to a wrapper that calls another BPDN-type solver. For general help with a specific function, type help then the function name for comments on its use.

2.0.1 Included Functions

The included functions are shown in Table 1. The main function is `rwl1sf_infer`.

For more information on these functions, please refer to their help files.

2.0.2 Writing Your Own Wrapper

While wrappers are included for the `l1ls` package's non-negative inference function, any function can be used as long as a wrapper is written for it and passed to the main function. To write a wrapper, simply use the same inputs/outputs as the existing wrappers and any extra parameters needed can be passed through using the `opts` struct. The inputs to the wrapper must be of the form `(dictionary_n, x_im, opts)`, and the output must be a single output: `coef_vals`. The wrapper simply extracts the necessary options from opts and organizes the inputs into the inference function.

References

[1] E. J. Candes and Y. Plan. Near-ideal model selection by ℓ_1 minimization. Technical report, California Institute of Technology, Dec 2007.

[2] E.J. Candès and M.B. Wakin. An introduction to compressive sampling. *IEEE Signal Processing Magazine*, 25(2):21–30, 2008.

[3] E.J. Candès, M.B. Wakin, and S.P. Boyd. Enhancing sparsity by reweighted ℓ_1 minimization. *Journal of Fourier Analysis and Applications*, 14(5):877–905, 2008.

[4] A. S. Charles and C. J. Rozell. Dynamic filtering of sparse signals using reweighted ℓ_1. *Proceedings of the ICASSP, Vancouver, Canada*, May 2013.

[5] A.S. Charles and C.J. Rozell. Spectral superresolution of hyperspectral imagery using reweighted ℓ_1 spatial filtering. *IEEE Geoscience and Remote Sensing Letters*, 11(3):602–606, March 2014.

[6] M. Elad, M.A.T. Figueiredo, and Y. Ma. On the role of sparse and redundant representations in image processing. *IEEE Proceedings - Special Issue on Applications of Compressive Sensing & Sparse Representation*, Oct 2008.

[7] S.-J. Kim, K. Koh, M. Lustig, S. Boyd, and D. Gorinevsky. An interior-point method for large scale l1-regularized least squares. *IEEE Journal on Selected Topics in Signal Processing*, 1(4):606–617, Dec 2007.

[8] D. Needell. Noisy signal recovery via iterative reweighted l1-minimization. In *Forty-Third Asilomar Conference on Signals, Systems and Computers*, pages 113–117, 2009.