RWL1-DF: Re-Weighted Dynamic Filtering for Time-Varying Signals

Adam S. Charles

August 21, 2012

Contents

1 Overview

This code package is an implementation of the basis pursuit de-noising dynamic filtering (BPDN-DF) described in [1] and the rw-weighted ℓ_1 dynamic filtering (RWL1-DF) algorithm described in [2]. The goal of these two algorithms is to causally estimate a time-varying sparse signal when an approximate model of the dynamics is known. The methodology assumes that the signal evolves as

$$\boldsymbol{x}_n = f_n(\boldsymbol{x}_{n-1}) + \boldsymbol{\nu}_n$$

where $\boldsymbol{x}_n \in \mathbb{R}^N$ is the evolving state, $f_n(\cdot)$ is the dynamics function and $\boldsymbol{\nu} \in \mathbb{R}^N$ is the innovations term, which represents the unpredictable portion of the dynamics. In this framework we assume that we obtain a number of measurements at each time-step

$$\boldsymbol{y}_n = \boldsymbol{G}_n \boldsymbol{x}_n + \boldsymbol{\epsilon}_n \tag{1}$$

where $\boldsymbol{y}_n \in \mathbb{R}^M$ is the measurement vector at each time step, $\boldsymbol{G}_n \in \mathbb{R}^{M \times N}$ is the linear measurement operator and $\boldsymbol{\epsilon}_n \in \mathbb{R}^M$ is the observation noise. We wish to recover the function $\boldsymbol{x}_n$ at each time-step, a problem commonly referred to as the *filtering* process. At each time step, we do not receive enough measurements to reconstruct the signal directly ($M \ll N$). Therefore knowledge of the dynamics has to be taken into account at each iteration. This code package contains two such algorithms as well as methods to compare to algorithms which do not take dynamic information into account (for comparison).

In particular the code package includes small-scale and large scale `MATLAB` implementations for both BPDN-DF and RWL1-DF. It also includes scripts and supporting functions to test the algorithms on a small-scale toy problem and a large-scale video recovery test. The included functions are outlined in Section 3.

2 Filtering Algorithms Included

2.1 Basis Pursuit De-noising Dynamic Filtering (BPDN-DF)

In basis pursuit de-noising (BPDN), a sparse signal is recovered via a convex optimization function [3, 4]. Specifically if $\boldsymbol{x}_n$ is sparse in some generative dictionary (often taken to be a wavelet basis) $\boldsymbol{W}$ (i.e. $\boldsymbol{a}_n$ is a sparse vector and $\boldsymbol{x}_n = \boldsymbol{W}\boldsymbol{a}_n$). BPDN can be used at each iteration to solve for the state given the measurements $\boldsymbol{y}_n$ as

$$\widehat{\boldsymbol{a}}_n = \arg\min_{\boldsymbol{a}} \|\boldsymbol{y}_n - \boldsymbol{G}_n \boldsymbol{W}\boldsymbol{a}\|_2^2 + \gamma \|\boldsymbol{a}\|_1$$

where γ trades off between sparsity and data fidelity, and the state is recovered by $\widehat{\boldsymbol{x}}_n = \boldsymbol{W}\widehat{\boldsymbol{a}}_n$. This optimization function, however, ignores inter-frame correlations. One method to induce correlations which is included in this package, is the BPDN dynamic filtering (BPDN-DF) [1]. In BPDN-DF we add another term to the optimization to include out knowledge of the dynamic process into the estimation process:

$$\widehat{\boldsymbol{a}}_n = \arg\min_{\boldsymbol{a}} \|\boldsymbol{y}_n - \boldsymbol{G}_n \boldsymbol{W}\boldsymbol{a}\|_2^2 + \gamma \|\boldsymbol{a}\|_1 + \kappa \|\boldsymbol{a} - \boldsymbol{a}_{est}\|_q^q$$

where κ is a second trade-off parameter which balances the prediction error to the sparsity and measurement fidelity terms and q is our choice of regularization norm. The choice in q depends on the statistics expected from the error in our assumed dynamic process (the innovations term). Gaussian innovations should result in $q = 2$ while sparse innovations should be included as $q \leq 1$.

2.2 Re-Weighted ℓ_1 Dynamic Filtering (RWL1-DF)

Recent expansions of the traditional BPDN framework has resulted in algorithms with improved performance in under-sampled signal recovery. In particular, the re-weighted ℓ_1 (RWL1) algorithm has provided improved recovery at the cost of solving a number of BPDN problems [5–8]. In the RWL1 algorithm, each element of the sparse decomposition $\boldsymbol{a}_n[i]$ is given its own weighting parameter $\boldsymbol{\lambda}_n[i]$. The weights are adjusted at each algorithmic step to further encourage coefficients that appear to be active while further discouraging coefficients which have little evidence. The algorithm iterates through

$$\widehat{\boldsymbol{a}}_n^t = \arg\min_{\boldsymbol{a}} \|\boldsymbol{y}_n - \boldsymbol{G}_n \boldsymbol{W} \boldsymbol{a}\|_2^2 + \lambda_0 \sum \boldsymbol{\lambda}_n^t[i]|\boldsymbol{a}[i]|$$
$$\widehat{\boldsymbol{\lambda}}_n^{t+1}[i] = \frac{\tau}{|\widehat{\boldsymbol{a}}_n^t[i]| + \eta}$$

where λ_0, τ and η are algorithmic constants that arise from the probabilistic model and t counts the algorithmic iterations (see [8] for more details). Again, this algorithm does not account for inter-frame correlations, so a dynamic filtering version of the RWL1 algorithm (RWL1-DF) was devised to address this issue [2]. In the RWL1-DF algorithm, the updates for the weights include a prediction for the weights

$$\boldsymbol{W}^T f_n\left(\boldsymbol{W}\widehat{\boldsymbol{a}}_{n-1}\right)$$

as

$$\widehat{\boldsymbol{a}}_n^t = \arg\min_{\boldsymbol{a}} \|\boldsymbol{y}_n - \boldsymbol{G}_n \boldsymbol{W} \boldsymbol{a}\|_2^2 + \lambda_0 \sum \boldsymbol{\lambda}_n^t[i]|\boldsymbol{a}[i]|$$
$$\widehat{\boldsymbol{\lambda}}_n^{t+1}[i] = \frac{\tau}{\beta|\widehat{\boldsymbol{a}}_n^t[i]| + |\boldsymbol{W}^T f_n\left(\boldsymbol{W}\widehat{\boldsymbol{a}}_{n-1}\right)| + \eta}$$

where λ_0, τ β and η are again algorithmic constants that arise from the probabilistic model and t counts the algorithmic iterations. This algorithms tends to be more robust to model mismatch than BPDN-DF, as some of the scripts included in this package show. More details and the full probabilistic model can be found in [2].

3 Code Included

3.1 Function Descriptions

Table 1: Test Scripts Included

Script Name	Description
`RWL1_DF_test.m`	Script to simulate and recover a simple, small time varying signal.
`RWL1_DF_testimage.m`	Script to import, compressively measure, and recover the foreman video sequence.

Table 2: Small Scale Function Implementations

Function Name	Description
`BPDN_multi.m`	Function to run BPDN on a series of signals
`BPDN_DF_L2L1.m`	Function to run BPDN-DF for sparse signals and Gaussian innovations
`BPDN_DF_L1.m`	Function to run BPDN-DF for Gaussian signals and Sparse innovations
`BPDN_DF_L1L1.m`	Function to run BPDN-DF for sparse signals and innovations
`RWL1_DF.m`	Function to run the RWL1-DF algorithm
`Basic_KF.m`	Function to run the Kalman Filter

Table 3: Large Scale Function Implementations

Function Name	Description
`BPDN_video.m`	Function to run BPDN on video sized data (implicit functions)
`BPDN_DF_video.m`	Function to run L1L2 BPDN-DF on video sized data
`RWL1_video.m`	Function to run RWL1 on video sized data
`RWL1_DF_largescale.m`	Function to run RWL1-DF on large data sets (implicit functions)

The main scripts (`RWL1_DF_test.m` and `RWL1_DF_imagetest.m`) give examples on how to run the included functions for small scale and large scale problems. In particular, `RWL1_DF_test.m` generates a random sequence of sparse vectors (which evolve according to a sequence of random permutations) and have an innovations with a preset mean sparsity. The sequence is recovered from compressed measurements (Gaussian random measurements) using standard Kalman Filtering, BPDN, BPDN-DF (three ways), RWL1 and RWL1-DF. In `RWL1_DF_imagetest.m`, the foreman video sequence is loaded and recovered from random subsamples of its noiselet transform (randomly sampled on a frame-by-frame basis). The video sequence is recovered using BPDN, BPDN-DF (with an ℓ_2 dynamics constraint), RWL1 and RWL1-DF. The functions used in `RWL1_DF_imagetest.m` all take implicit functions in the form of a cell array for both the dynamics functions (in this case taken to be identity) and the measurement functions (sub-sampled noiselet transforms in this case). For more assistance on each function (e.g. inputs/outputs), please refer to the help files included.

Table 4: Supporting Functions

Function Name	Description
noise_model.m	Simulates a sparse state noise model with random deviations from some true dynamics
RandPermMat.m	Generates a random permutation matrix
rand_posn.m	Generates random Poisson variable
rand_sparse_seq.m	Generates a random sparse permutation sequence with sparse deviations
opt_L1_dyn1.m	Custom function for minFunc to calculate BPDN-DF estimates with sparse states and innovations

3.2 Package Dependencies

Table 5: Code Dependencies

Package	Location	Dependencies
L1 Homotopy [9]	http://users.ece.gatech.edu/~sasif/homotopy/index.html	BPDN_multi.m, BPDN_DF_L2L1.m, BPDN_DF_L1.m, RWL1_DF.m
TFOCS [10]	http://tfocs.stanford.edu/download/	BPDN_video.m, BPDN_DF_video.m, RWL1_video.m, RWL1_DF_Video.m
minFunc	http://www.di.ens.fr/~mschmidt/Software/minFunc.html	BPDN_DF_L1L1.m
CS-Mag Code [11]	http://users.ece.gatech.edu/~justin/spmag/	RWL1_DF_testimage.m (optional)
WaveLab	http://www-stat.stanford.edu/~wavelab/	RWL1_DF_testimage.m (optional)
DT-DWT [12]	http://eeweb.poly.edu/iselesni/WaveletSoftware/	RWL1_DF_testimage.m (optional)

There are a number of other packages which the code in this package depends on. In particular, two ℓ_1 optimization solvers are required (one for small scale problems and one for larger scale problems), a generalized optimization routine, and two packages are needed to efficiently calculate different transforms. While most of these packages are optional (e.g. the wavelet packages can easily be swapped for other wavelet transforms) they are all necessary to run the main scripts included. See Table 5 for a full list of packages, where to obtain them, which functions require them and any possible related publications.

References

[1] A. Charles, M. Asif, J. Romberg, and C. Rozell, "Sparsity penalties in dynamical system estimation," in *Information Sciences and Systems (CISS), 2011 45th Annual Conference on*. IEEE, pp. 1–6.

[2] A. Charles and C. Rozell, "Re-weighted ℓ_1 dynamic filtering for time-varying sparse signal estimation," 2012, submitted.

[3] R. Baraniuk, "Compressive sensing," *IEEE Signal Processing Magazine*, vol. 24, no. 4, pp. 118–121, Jul 2007.

[4] E. Candes, J. Romberg, and T. Tao, "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information," *IEEE Trans on Information Theory*, vol. 52, no. 2, Feb 2006.

[5] E. Candes, M. B. Wakin, and S. P. Boyd, "Enhancing sparsity by reweighted ℓ_1 minimization," *Journal of Fourier Analysis and Applications*, vol. 14, no. 5, pp. 877–905, Dec 2008, special Issue on Sparsity.

[6] D. Wipf and S. Nagarajan, "Iterative reweighted ℓ_1 and ℓ_2 methods for finding sparse solutions," *Selected Topics in Signal Processing, IEEE Journal of*, vol. 4, no. 2, pp. 317–329, 2010.

[7] R. Chartrand and W. Yin, "Iteratively reweighted algorithms for compressive sensing," in *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*. IEEE, 2008, pp. 3869–3872.

[8] P. Garrigues and B. Olshausen, "Group sparse coding with a laplacian scale mixture prior," *Advances in Neural Information Processing Systems*, pp. 1–9, 2010.

[9] M. Salman Asif and J. Romberg, "Dynamic updating for minimization," *IEEE Journal of Selected Topics in Signal Processing*, vol. 4, no. 2, pp. 421–434, 2010.

[10] S. Becker, E. Candès, and M. Grant, "Templates for convex cone problems with applications to sparse signal recovery," *Mathematical Programming Computation*, pp. 1–54, 2011.

[11] J. Romberg, "Imaging via compressive sampling," *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 14–20, 2008.

[12] I. Selesnick, "The double-density dual-tree dwt," *Signal Processing, IEEE Transactions on*, vol. 52, no. 5, pp. 1304–1314, 2004.