Math 76 Project

Anne Gelb
Department of Mathematics
Dartmouth College

June 23, 2017

1 Introduction

This project uses the results of the following papers:

1. D. Denker and A. Gelb, *Edge Detection of Piecewise Smooth Functions from Under-Sampled Fourier Data Using Variance Signatures*, SIAM Journal on Scientific Computing, **39:2** 559-592 (2017).

2. R. Archibald, A. Gelb, and R. Platte, *Image Reconstruction from Undersampled Fourier Data Using the Polynomial Annihilation Transform*, Journal of Scientific Computing, **67** 432–452, (2016), DOI 10.1007/s10915-015-0088-2.

3. W. Stefan, R. Renaut and A. Gelb *Improved Total Variation-type Regularization Using Higher-order Edge Detectors*, SIAM Journal on Imaging Sciences, **3:2** 232-251 (2010).

4. R. Archibald, A. Gelb, and J. Yoon, *Polynomial Fitting for Edge Detection in Irregularly Sampled Signals and Images*, SIAM Journal on Numerical Analysis, **43**, 259–279 (2005).

5. G. Wasserman, R. Archibald and A. Gelb, *Image Reconstruction from Fourier Data Using Sparsity of Edges*, Journal of Scientific Computing, **65:2** 533–552, (2015).

2 Problem set up

Let $f(x)$ be a piecewise smooth function on an interval $[-1, 1]$. Assume we have M vectors $\mathbf{g_1}, \cdots, \mathbf{g_M}$. Each $\mathbf{g}_m$ approximates $f(x)$ at a set of grid points $x_j = -1 + \frac{2j}{N}$, $j = 0, \cdots, N$. We can assume that f is periodic, although in general this is not needed. Each entry of $\mathbf{g}_m$ is given by $f(x_j) + \epsilon_j$, where ϵ_j is white noise. Part of this experiment will be to see how robust our algorithm will be with respect to signal to noise ratio (SNR).

Our overall goal is to recover the true function $f(x)$ on any set of grid ponts (we can use the same grid points x_j for now). We will break this up into different tasks. **Part 4** and **Part 5** are optional. Groups can work on **Part 1** and **Part 2** together, and then split off to work on various aspects of **Part 3**, **Part 4**, and **Part 5**. As a **Part 6**, you can come up with a completely different application, as long as you can justify it in some real world scenario. (There is actually another variation on this problem that will be described in **Project 2**, which I will be posting soon.)

I encourage you to consult each other when working on this project, especially in terms of programming. Remember, everyone is coming into this with different backgrounds, so you will learn a lot from talking together. Don't hesitate to use canned packages, no one should reinvent the wheel!

- **Part 1:** First assume that $M = 1$. Use the *polynomial annihilation method* (see Archibald, Gelb, and Yoon) to recover the *jump function* $[f](x)$ of piecewise smooth function $f(x)$ at grid points x_j. You should choose several different piecewise smooth functions, with varying amounts of noise (SNR) and resolution (N). You should also choose different *orders* of he polynomial annihilation method. Compare different results. When does the method work well? What causes the method to break down?

- **Part 2:** We still assume that $M = 1$. In (Archibald, Gelb, and Platte) and (Wasserman, Archibald, and Gelb) the polynomial annihilation method is used as an ℓ_1 regularization transform operator. In these papers, the given data are assumed to be Fourier samples, $\hat{\mathbf{f}}$, so that the solution $\mathbf{f}$ should minimize $||\mathcal{F}\mathbf{f} - \hat{\mathbf{f}}||$ where F is the Fourier transform matrix. Here, however, we assume we are given grid point data so we use $F = I$, the identity matrix, and the given data are given by $\mathbf{g}$ above. (A similar framework is in (Stefan, Renaut and Gelb).) Use the results from **Part 1** to recover the underlying function f at the grid points x_j, $j = 0, \cdots, N$.

- **Part 3:** Now starts the new stuff! Suppose now $M > 1$ (you will vary M as part of the project). Because each vector is supposed to measure the same function, f, we can assume that the data set $\{\mathbf{g}_m\}_{m=1}^M$ should have features in common. In particular, they should all have the same set of *edges*, and since the underlying piecewise smooth there are only a finite number of edges, that is, the function f has sparse representation in the edge domain. This is the concept used for $M = 1$ as well, but here we are going to see if there is an advantage to using *joint sparsity*, that is, these collected data sets are highly correlated in their sparsity domain. Some questions to ask: How large should M be for it to be useful? If you have large M, how small can N be? Is there an optimal choice for M and N in terms of accuracy and efficiency? How do M and N vary with the amount of noise? How robust is the method to noise? Is there a better way to choose the parameters (e.g. weights and optimization paramaters) for the minimization scheme? Below I sketch some useful ideas. It is up to you to see what to do.

- **Part 4:** Applications: This is some modification of the cocktail party problem. You have several different audio recordings of some conversations, and you (perhaps you are a spy!) want to here one particular conversation. This conversation is common to all of the audio recordings, but there could be a lot of noise in each of them. Come up with an application (a simplified version) and try out this technique. Observe that you can actually use *different* data collection modalities, for example perhaps $\mathbf{g}_1$ is Fourier data of f while $\mathbf{g}_2$ is pixel data, as above.

- **Part 5:** Applications: Another problem is that the data $\mathbf{g}$ is blurry, that is $\mathbf{g} = \mathbf{k} * \mathbf{f} + \epsilon$ where k is a blur operator and $*$ means convolution. When data are blurry, it can become more difficult to recover edges. Do the multiple vector measurements help in this case?

3 Background information for multiple measurement problem

Let $\boldsymbol{f} \in \mathbb{R}^N$ denote a vector (discrete signal) to be recovered. We denote the objects to be recovered by $\boldsymbol{f}_1, \ldots, \boldsymbol{f}_M$, where $M \in \mathbb{N}$ is the total number of objects. We can write

$$\boldsymbol{F} = [\boldsymbol{f}_1 | \cdots | \boldsymbol{f}_M] \in \mathbb{R}^{N \times M},$$

Measurements of the vectors $\boldsymbol{f}_1, \dots, \boldsymbol{f}_M$ are taken according to measurement matrices $\boldsymbol{A}_1, \dots, \boldsymbol{A}_M \in \mathbb{R}^{M \times N}$, giving measurements

$$\boldsymbol{g}_m = \boldsymbol{A}_m \boldsymbol{f}_m + \boldsymbol{\epsilon}_m, \qquad m = 1, \dots, M. \tag{3.1}$$

Here $\boldsymbol{\epsilon}_1, \dots, \boldsymbol{\epsilon}_M$ are noise vectors. Often it will be the case that

$$\boldsymbol{A} = \boldsymbol{A}_1 = \cdots = \boldsymbol{A}_M,$$

although this condition is not necessary for the technique. (In our sample problem $A_m = I$, the identity matrix, for each $m = 1, \cdots, M$.) In this latter case, note that we may rewrite (3.1) as

$$\boldsymbol{G} = \boldsymbol{A}\boldsymbol{F} + \boldsymbol{\Omega},$$

where

$$\boldsymbol{G} = [\boldsymbol{g}_1 | \cdots | \boldsymbol{g}_M] \in \mathbb{R}^{N \times M}, \qquad \boldsymbol{\Omega} = [\boldsymbol{\epsilon}_1 | \cdots | \boldsymbol{\epsilon}_M] \in \mathbb{R}^{N \times M}.$$

We write $\|\cdot\|_p$ for the ℓ^p-norm on $\mathbb{R}^N$, where $p \geq 1$. As is conventional, we write $\|\cdot\|_0$ for the ℓ^0-norm, i.e.

$$\|\boldsymbol{h}\|_0 = |\mathrm{supp}(\boldsymbol{h})|.$$

Here $\mathrm{supp}(\boldsymbol{h})$ is the support of $\boldsymbol{h}$, defined by

$$\mathrm{supp}(\boldsymbol{h}) = \{i : h_i \neq 0\}, \qquad \boldsymbol{h} = (h_i)_{i=1}^N.$$

Given a vector $\boldsymbol{w} = (w_i)_{i=1}^N$ of positive weights, we define the weighted $\ell^1_{\boldsymbol{w}}$-norm as

$$\|\boldsymbol{h}\|_{1,\boldsymbol{w}} = \sum_{i=1}^N w_i |h_i|.$$

One can also define weighted $\ell^p_{\boldsymbol{w}}$-norms for $p \neq 1$, however this shall not be needed for the paper.

If $\boldsymbol{H} = (h_{im})_{i,m=1}^{N,M} \in \mathbb{R}^{N \times M}$ is a matrix, we define the $\ell^{p,q}$-norms by

$$\|\boldsymbol{H}\|_{p,q} = \left(\sum_{i=1}^N \left(\sum_{m=1}^M |h_{im}|^q \right)^{p/q} \right)^{1/p}.$$

In particular, if $p = q = 2$ then

$$\|\boldsymbol{H}\|_{2,2} = \|\boldsymbol{H}\|_F = \left(\sum_{i=1}^N \sum_{m=1}^M |h_{im}|^2 \right)^{1/2},$$

is just the Frobenius norm of $\boldsymbol{H}$. We also define the weighted $\ell^{1,2}_{\boldsymbol{w}}$-norm by

$$\|\boldsymbol{H}\|_{1,2,\boldsymbol{w}} = \sum_{i=1}^N w_i \left(\sum_{m=1}^M |h_{im}|^2 \right)^{1/2}.$$

4 Description of the variance joint sparsity ℓ_1 method

Let $\boldsymbol{f}_1, \ldots, \boldsymbol{f}_M \in \mathbb{R}^N$ be the sequence of vectors to recover and

$$\boldsymbol{g}_m = \boldsymbol{A}_m \boldsymbol{f}_m + \boldsymbol{\epsilon}_m, \qquad m = 1, \ldots, M,$$

the vector of measurements. We shall assume the *a priori* noise bound

$$\|\boldsymbol{\epsilon}_m\|_2 \leq \eta_m, \qquad m = 1, \ldots, M,$$

for known noise levels $\eta_1, \ldots, \eta_M$.

For the variance joint sparsity ℓ_1 (VJSL1) method, we shall make the following assumptions:

1. The supports of the vectors are similar, i.e. $\operatorname{supp}(\boldsymbol{h}_1) \approx \operatorname{supp}(\boldsymbol{h}_2) \approx \cdots \approx \operatorname{supp}(\boldsymbol{h}_M)$, where h is a measureable feature of f (e.g. the jump function $[\mathbf{f}] \approx \mathbf{Pf} = \mathbf{h}$).

2. The coefficients of the vectors are sufficiently distinct. Specifically, the vector $\boldsymbol{v} = (v_i)_{i=1}^N$ of pixelwise variances

$$v_i = \frac{1}{M} \sum_{m=1}^{M} (h_{im})^2 - \left(\frac{1}{M} \sum_{m=1}^{M} h_{im} \right)^2, \qquad i = 1, \ldots, N,$$

   is nonzero with $\operatorname{supp}(v) \approx \bigcup_{m=1}^{M} \operatorname{supp}(\boldsymbol{h}_m)$.

We note in passing that VJSL1 will not fail when either of these assumptions does not hold. Rather, it will just not convey any benefit over individual recovery of the $\boldsymbol{h}_m$. The first assumption is essentially necessary for any technique that seeks to exploit joint sparsity.

The VJSL1 method now proceeds in four steps:

Step 1: In the first step, we recover the vectors separately using standard ℓ^1 minimization:

$$\check{\boldsymbol{f}}_m \in \underset{\boldsymbol{z} \in \mathbb{R}^N}{\operatorname{argmin}} \|\boldsymbol{P}\boldsymbol{z}\|_1 \text{ subject to } \|\boldsymbol{A}_m \boldsymbol{z} - \boldsymbol{g}_m\|_2 \leq \eta_m, \qquad m = 1, \ldots, M.$$

Step 2: In the second step, we compute the pixel-wise variance of the vectors $\check{\boldsymbol{h}}_m = (\check{h}_{im})_{i=1}^N$, $m = 1, \ldots, M$, computed in the first step. That is, we compute $\check{\boldsymbol{v}} = (\check{v}_i)_{i=1}^N$, where

$$\check{v}_i = \frac{1}{M} \sum_{m=1}^{M} (\check{h}_{im})^2 - \left(\frac{1}{M} \sum_{m=1}^{M} \check{h}_{im} \right)^2, \qquad i = 1, \ldots, N.$$

Step 3: The two assumptions made above suggest that $\boldsymbol{v}$ should carry information about the shared support of the $\boldsymbol{h}_m$. Specifically, $\check{v}_i$ should be large when the index i belongs to this support, and $\check{v}_i \approx 0$ otherwise. In the third step, we compute a vector of nonnegative weights $\boldsymbol{w} = (w_i)_{i=1}^N$ based on this information, where $0 \leq w_i \leq 1$. In particular, we choose $w_i \approx 0$ when $\check{v}_i$ is large and $w_i \approx 1$ when $\check{v}_i \approx 0$.

Step 4: Having defined the vector $\boldsymbol{w}$, we in the final step we solve M weighted ℓ^1 minimization problems to get the final reconstruction of the vectors $\boldsymbol{f}_m$:

$$\hat{\boldsymbol{f}}_m \in \underset{\boldsymbol{z} \in \mathbb{R}^N}{\operatorname{argmin}} \|\boldsymbol{P}\boldsymbol{z}\|_{1,\boldsymbol{w}} \text{ subject to } \|\boldsymbol{A}_m \boldsymbol{z} - \boldsymbol{g}_m\|_2 \leq \eta_m, \qquad m = 1, \ldots, M.$$

Note that this method is *parallelizable*, since the computationally-intensive steps (Steps 1 & 4) each require the solution of M separate (weighted) ℓ^1-minimization problems. Steps 2 & 3 are require communications between cores, but are extremely cheap in comparison.

4.1 Choice of weights

In order to implement VJSL1, one needs to specify the weights $\boldsymbol{w}$. One possible option is to use *reciprocal weights*, which is given as follows:

Fix a parameter $\epsilon > 0$, and define the weights by

$$w_i = \frac{1}{v_i + \epsilon}, \qquad i = 1, \ldots, N.$$

This strategy can adapt to differing scales in the variance vector $\boldsymbol{v}$.