Applications to Data Smoothing and Image Processing I

MA 348

Kurt Bryan

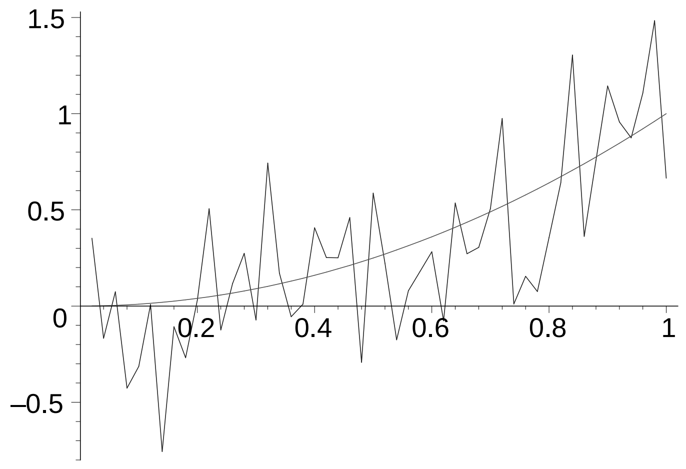
Signals and Images

Let t denote time and consider a signal $a(t)$ on some time interval, say $0 \leq t \leq 1$. We'll assume that the signal $a(t)$ is a continuous or even differentiable function of time t.

Of course when you actually measure such a signal you measure it only at discrete times $t_1, t_2, \ldots, t_n$; let $a_i = a(t_i)$ for $1 \leq i \leq n$. Such measurements invariably contain noise, so what we actually obtain is not a_i, but rather $y_i = a_i + \epsilon_i$ where ϵ_i denotes noise or measurement error. Our goal is to recover the a_i given measurements y_i. Of course this seems impossible. It is.

But we can do something if we're willing to make some a priori assumptions about the nature of $a(t)$ and the noise ϵ_i. Specifically, we will assume (for the moment) that $a(t)$ is differentiable and that all ϵ_i are identically distributed independent random variables (but not necessarily normal). This is a pretty typical model for random noise (but not the only one!)

Here's a picture of the situation: Let $t_i = i/n$ for $1 \leq i \leq n$, with $n = 50$ and $a(t) = t^2$. I take each ϵ_i to be independent and normally distributed with zero mean and standard deviation 0.3. A plot of both the x_i and $y_i = a_i + \epsilon_i$ is shown below:



Recovering the underlying "true" signal from the noisy version will be tough. What we've got going for us is this: The underlying signal is assumed to be smooth and the noise is "rough". This gives us a way to try removing the noise without messing up the signal.

Consider recovering the a_i by minimizing the function

$$f(\mathbf{x}) = \frac{1}{2}\sum_{i=1}^{n}(x_i - y_i)^2.$$

The vector $\mathbf{x} = [x_1, x_2, \ldots, x_n]$ will represent our best estimate of the underlying signal. Of course the minimum occurs at $x_i = y_i = a_i + \epsilon_i$, so this approach doesn't remove any noise! Let's add an additional term to the objective function f to form

$$f(\mathbf{x}) = \frac{1}{2}\sum_{i=1}^{n}(x_i - y_i)^2 + \frac{\alpha}{2}\sum_{i=1}^{n-1}\left(\frac{x_{i+1} - x_i}{h}\right)^2 \tag{1}$$

where α is to be determined. The claim is that by minimizing $f(\mathbf{x})$ as defined by equation (1), we recover a "noise-reduced" signal, at least if we choose α intelligently.

The reason is this: In the minimizing process the first term $\sum_{i=1}^{n}(x_i - y_i)^2$ will try to force $x_i \approx y_i$. But since h is small then the second term on the right in (1) encourages taking x_{i+1} fairly close to x_i, i.e., encourages a "smooth" reconstruction in which the x_i don't change too rapidly, in contrast to the noisy y_i. The second term involving α is a *penalty term*, of which we'll see more examples later. In this case this term penalizes estimates x_i which vary too rapidly. Taking $\alpha = 0$ imposes no penalty, and the reconstructed signal is just the y_i.

Finding the minimum of $f(\mathbf{x})$ is in principle easy. The function is quadratic, so the normal equations will be linear. Differentiate with respect to each variable x_j to find

$$\frac{\partial f}{\partial x_j} = -\frac{\alpha}{h^2}x_{j+1} + (1 + 2\frac{\alpha}{h^2})x_j - \frac{\alpha}{h^2}x_{j-1} \tag{2}$$

for $1 < j < n$, while $\frac{\partial f}{\partial x_1} = (1 + \frac{\alpha}{h^2})x_j - \frac{\alpha}{h^2}x_2$ and $\frac{\partial f}{\partial x_n} = (1 + \frac{\alpha}{h^2})x_n - \frac{\alpha}{h^2}x_{n-1}$. These equations become $\mathbf{Ax} = \mathbf{a}$ where $\mathbf{A} = \mathbf{I} + \frac{\alpha}{h^2}\mathbf{B}$ with

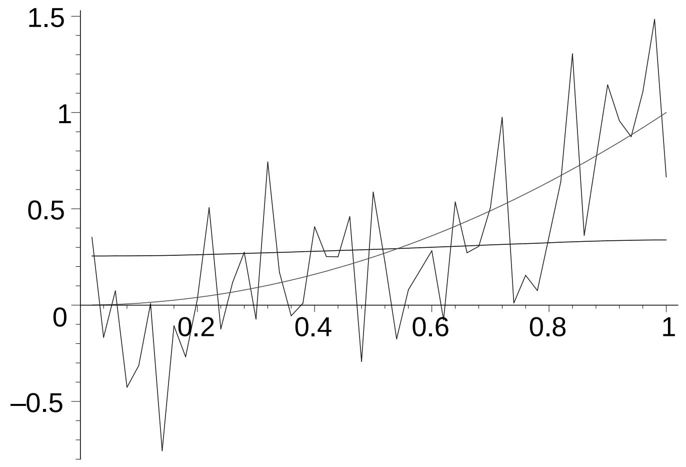
$$\mathbf{B} = \begin{bmatrix} 1 & -1 & 0 & 0 & \cdots & 0 \\ -1 & 2 & -1 & 0 & \cdots & 0 \\ 0 & -1 & 2 & -1 & \cdots & 0 \\ & & & & \vdots & \\ 0 & \cdots & 0 & -1 & 2 & -1 \\ 0 & \cdots & 0 & 0 & -1 & 1 \end{bmatrix}.$$

and $\mathbf{a} = [a_1, a_2, \ldots, a_n]^T$. This is a large SPARSE matrix. An excellent method for solving is conjugate gradients, which of course involves minimizing the original f. So we really didn't need to derive the normal equations (I just wanted you to see an example where a big linear system is equivalent to minimizing a certain function). What we're really doing is minimizing $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} - \mathbf{x}^T\mathbf{a}$.

It also turns out that $\mathbf{B}$ is positive semi-definite (see if you can prove this. Hint: expand out $\mathbf{x}^T\mathbf{B}\mathbf{x}$) so that $\mathbf{A} = \mathbf{I} + \frac{\alpha}{h^2}\mathbf{B}$ will be positive definite if $\alpha \geq 0$. And $\mathbf{A}$ is obviously symmetric.

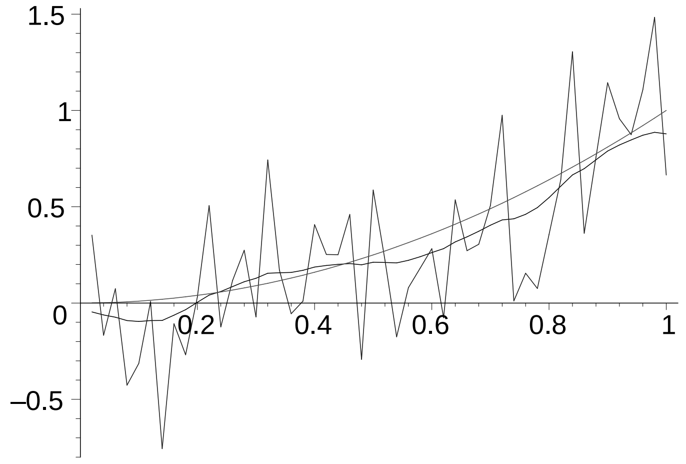
Numerical Experiments

As mentioned, taking $\alpha = 0$ will simply return the noisy sampled data as the smoothed signal. That's not interesting. Using the noisy data in the above figure, I took $\alpha = 1$ and used conjugate gradients to minimize f. Now note that $\alpha = 1$ means the coefficient in the penalty term is effectively $\alpha/h^2 = 2500$, pretty big. The resulting smoothed signal looks like a horizontal line!

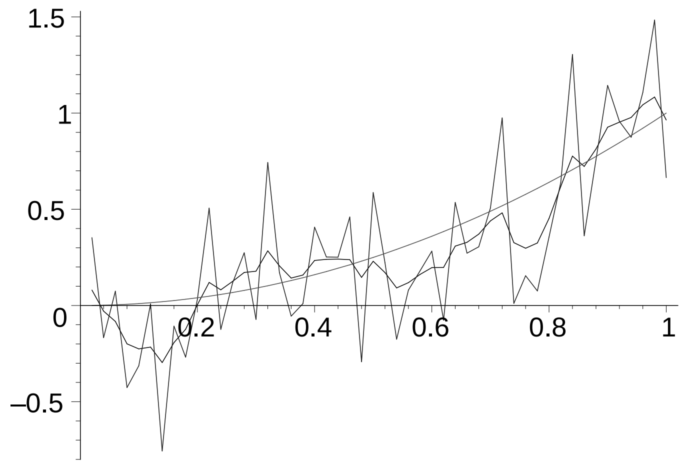


The penalty is way too high, biased too heavily in favor of smoothing.

Taking $\alpha = 0.01$ yields a pretty good reconstruction:



Dropping α to 0.001 gives



Now the noise, although smoothed, is more prominent.

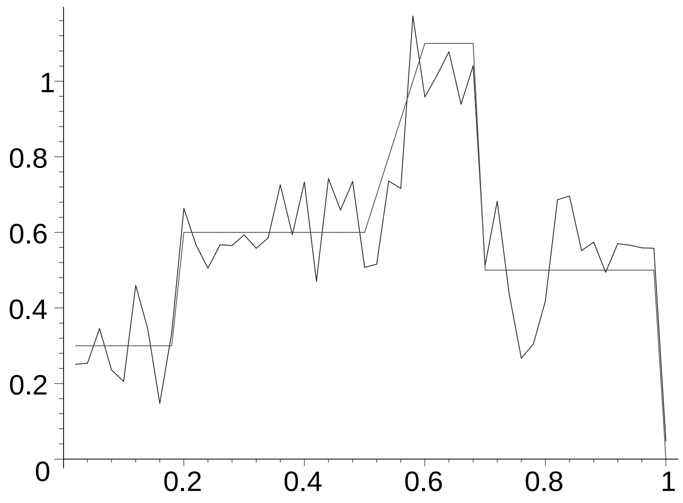
Blocky Signals and Images

The ideas above generalize easily to reconstructing two dimensional signals, i.e., images. A two dimensional signal or image on the unit square in $\mathbb{R}^2$ might be modelled as $a(s,t)$, where I'm using s and t as 2D coordinates. This assumes that nature of the image at any point can be represented by a single number, e.g., a grey-scale image. A sample of this signal might look like $a_{ij} = a(s_i, t_j)$ where $s_i = i/n$, $t_j = j/n$ for $1 \leq i, j \leq n$. Our sample of the signal would be something like $y_{ij} = a_{ij} + \epsilon_{ij}$ where the ϵ_{ij} are random and independent.

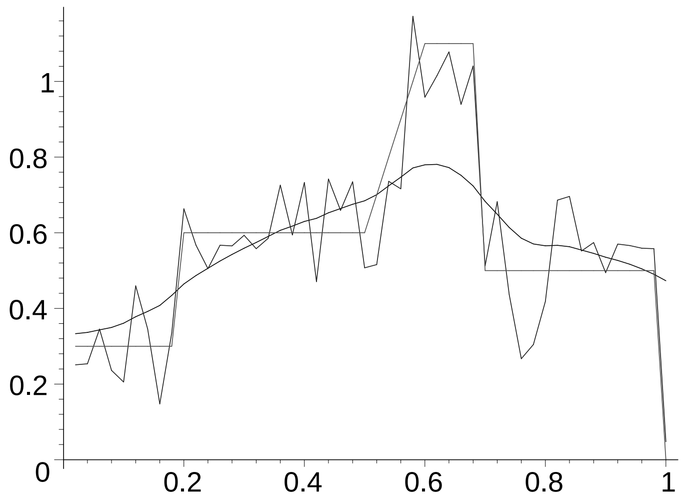
We can use the same approach as before. We construct a function f which contains a sum of the form $\sum_{ij}(x_{ij} - y_{ij})^2$ plus a penalty term. The penalty term can take many forms, but one idea is to penalize any rapid change in the x or y directions. Thus the penalty term might contain terms like $\frac{(x_{i+1,j}-x_{i,j})^2}{h^2}$, which penalizes x change, and $\frac{(x_{i,j+1}-x_{i,j})^2}{h^2}$, which penalizes y change. We could then smooth or "de-noise" 2D signals in the same manner.

But there is a problem with this formulation. Images, even when free of noise, are NOT usually smooth signals—think of a typical photograph. Images usually consists of many "homogeneous" regions in which parameters vary little, separated by sharp transitions or edges. The scheme above, and in particular the quadratic penalty term, doesn't like these transitions and tries to smooth them out, and so excessively blurs the image.

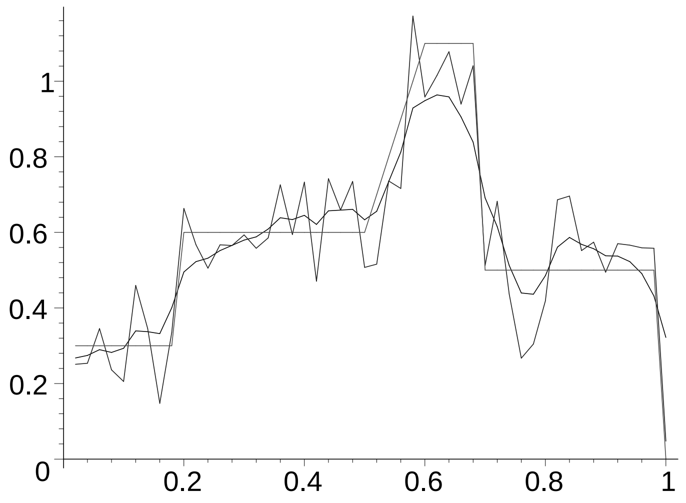
This is best seen using a 1D example. Consider the following one-dimensional "image", with a noisy version superimposed:



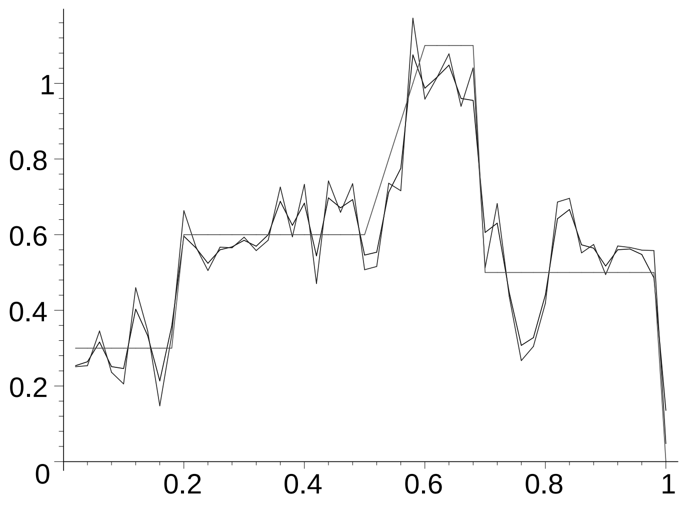
The true signal is only piecewise smooth, with a few sharp transitions or edges between smooth regions. If you saw the smooth signal there'd be little doubt it's uncorrupted by noise. That's what we want to recover from the noisy signal. But here's the de-noised version using $\alpha = 0.01$ with a squared penalty term like in (1):



The edges of the image have been trashed! Decreasing α to 0.001 produces



which is better. With $\alpha = 0.0001$ we get



in which the reconstructed smoothed signal now pretty much resembles the noisy signal. There's no suitable value for α in which we can obtain a relatively noise free image that retains the edges that make up a good image.

Ideas for a Fix

What we really want to eliminate in a reconstructed image isn't change, but unnecessary change. We want to allow clean "jumps", but penalize unnecessary up and down "jaggedness". Consider the line $x(t) = t/2$ for $0 \le t \le 1$. The function rises from 0 to 1/2. We don't want to (excessively) penalize this kind of simple change, especially if it happens rapidly (say, if the 1/2 rise occurs in a t interval of length 0.01). What we want to penalize is a function like $x(t) = t/2 + \sin(100\pi t)$, which also changes from 0 to 1/2, but in a very oscillatory manner.

One way to do this is to change the penalty term to use absolute values instead of squaring, so in the 1D case as described above we consider the objective function

$$f(\mathbf{x}) = \frac{1}{2}\sum_{i=1}^{n}(x_i - y_i)^2 + \frac{\alpha}{2}\sum_{i=1}^{n-1}\frac{|x_{i+1} - x_i|}{h}. \tag{3}$$

Exercise

- Let $x(t) = t/2$ and $x_i = x(i/100)$ for $1 \le i \le 100$. Note that $x(t)$ rises from 0 to 1/2 on the interval $0 \le t \le 1$. Of course $h = 0.01$. Compute the penalty term on the right in both equations (1) and (3).

  Now let $x(t) = 50t$ and $x_i = x(i/10000)$ for $1 \le i \le 100$. Note that $x(t)$ rises from 0 to 1/2 on the interval $0 \le t \le 0.01$. Here $h = 0.0001$. Compute the penalty term on the right in both equations (1) and (3).

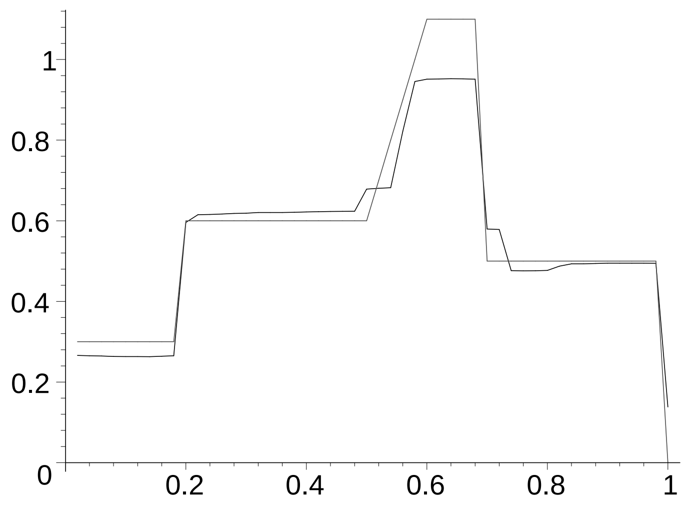
  Compare how each type of penalty treats the rise from 0 to 1/2.

The problem with minimizing f as defined by equation (3) is obvious: it's not differentiable. You could try an optimization algorithm that doesn't require differentiability. The obvious choice is Nelder-Mead, but on a problem of 50 or so variables this will be glacially slow.

Here's another idea: Let's replace the absolute value with a smooth function which is close to the absolute value. One nice choice is to take $\phi(x) = \sqrt{x^2 + \epsilon}$. This function is infinitely differentiable for $\epsilon > 0$, but if ϵ is small then $\phi(x) \approx |x|$. We'll thus minimize

$$f(\mathbf{x}) = \frac{1}{2}\sum_{i=1}^{n}(x_i - y_i)^2 + \frac{\alpha}{2}\sum_{i=1}^{n-1}\frac{\phi(x_{i+1} - x_i)}{h}. \tag{4}$$

This introduces another problem: if ϵ is very small then near the minimum the function f will have large second derivatives, and this confuses optimization algorithms (think about why, say in one dimension). So we'll start by taking a relatively large value for ϵ and minimizing f. Then we'll decrease ϵ and use the previous minimize as an initial guess for the smaller value of ϵ. A few repetitions of this can be an effective method for locating a suitable minimum. I set $\epsilon = 0.01$ and $\alpha = 0.01$, then minimized f using a conjugate gradient method. Then I set $\epsilon = 0.0001$ and used the $\epsilon = 0.01$ minimizer as a good initial guess. I then found the minimum for $\epsilon = 0.0001$ and used that as an initial guess for $\epsilon = 10^{-6}$. The result is (without the noisy signal overlayed)



That's a lot better. The noise has been damped out, but not at the expense of smoothing the edges in the signal.

Remarks

These ideas are currently hot topics in applied math and image reconstruction. In fact, if you let n (the number of sample points) tend to infinity you obtain a continuous version of the optimization problem which fits naturally into the mathematical framework provided by partial differential equations and the calculus of variations (a 200 year old subject which is basically infinite-dimensional optimization). We'll look at these areas briefly in a couple weeks.

If you're interested in the image restoration stuff, look at "Mathematical Problems in Image Processing" by Gilles Aubert and Pierre Kornprobst, Springer-Verlag, 2001.