Notes on Regularization[1]

David J. Fleet Allan Jepson

March 21, 2005

Regularization is a class of techniques that have been widely used to solve interpolation and approximation problems that frequently arise in image processing, computer vision, and computer graphics [1, 2, 4, 5, 6, 7]. The techniques are designed to find a "smooth", discrete signal v given a set of noisy and partial measurements u. For a one-dimensional signal, the relationship between v and u may be modeled by:

$$u[x] = v[x] + n[x], \quad x \in P \tag{1}$$

where n is a random field (e.g., a field of independent, zero mean, Gaussian random variables with common variance), and P is a subset of sample positions where the measurements u are available.

Problems such as these arise in many contexts:

- If v and u are images, then we have a denoising problem. That is, given a set of noisy image measurements, u, we wish to denoise it to recover v.
- If the measured signal, u, is subsampled and noisy, we might want to estimate the original image v on a denser sampling lattice. In this case we have many fewer measurements than we have unknowns.
- In image warping it is often the case that we are given a select set of points at which we know displacements from a warped image to coordinates in the original image. But often we want to compute a dense, smooth warp map from this set of sparse displacements so that we can deform an entire image. This is a form of interpolation/approximation problem.
- We may wish to interpolate a set of orientations measured at strong image edges to be used for the orientation of brush strokes in painterly rendering.

Denoising.

To start with a simple example, assume that the noisy measurements u are available at every pixel. We observe a noisy signal u and we wish to recover v. Of course, this problem as stated is underconstrained; we need to know something about the original signal v and/or the noise. A typical regularization solution assumes that the original signal is smooth and chooses v to minimize the following objective function:

$$E(v) = \sum_{x=1}^{N} (v[x] - u[x])^2 + \lambda \sum_{x=1}^{N-1} (v[x+1] - v[x])^2, \tag{2}$$

where N is the number of samples in the signal. The first term is called the data constraint, specifying that the solution v should be close to the measurements u. The second term is called the smoothness constraint, specifying that neighboring sample values of v should be similar. The parameter λ determines the tradeoff between the two constraints; if λ is large then the solution will be smoother at the expense of being further from the measurements. It is worth noting that

[1] Based on a handout by David Heeger at Stanford University.

we have (rather arbitrarily) adopted a particular definition of what "smoothness" means, a point that we will return to later. It is also worth noting that we have (arbitrarily) used quadratic error functions, another point that we will return to later.

Taking the derivative of Eq. 2 with respect to the k^{th} sample value $v[k]$ gives:

$$\begin{aligned}\frac{\partial E(v)}{\partial v[k]} &= \frac{\partial}{\partial v[k]}\left[(v[k]-u[k])^2+\lambda(v[k+1]-v[k])^2+\lambda(v[k]-v[k-1])^2\right] \\ &= 2(v[k]-u[k])+2\lambda(-v[k-1]+2v[k]-v[k+1]).\end{aligned}$$

Taking the derivatives for each sample and setting them equal to zero gives a set of linear equations of the form:

$$v[x]+\lambda(-v[x-1]+2v[x]-v[x+1])=u[x]. \tag{3}$$

This equation holds for all points x except the two end points, $x=1$ and $x=N$ because they only have one neighbor each. Therefore, it is important to consider them separately. If we take the derivatives of Eq. 2 with respect to $v[1]$ and $v[N]$, and set them equal to zero, then we get slightly different equations:

$$\begin{aligned}v[1]+\lambda(v[1]-v[2]) &= u[1] \\ v[N]+\lambda(v[N]-v[N-1]) &= u[N]\end{aligned}$$

Including the end points, we now have N linear equations that can be written collectively using matrix notation:

$$\begin{pmatrix} 1+\lambda & -\lambda & 0 & 0 & \dots & 0 \\ -\lambda & 1+2\lambda & -\lambda & 0 & \dots & 0 \\ 0 & -\lambda & 1+2\lambda & -\lambda & \dots & 0 \\ & & \ddots & \ddots & \ddots & \\ 0 & \dots & & 0 & -\lambda & 1+\lambda \end{pmatrix} \begin{pmatrix} v[1] \\ v[2] \\ v[3] \\ \vdots \\ v[N] \end{pmatrix} = \begin{pmatrix} u[1] \\ u[2] \\ u[3] \\ \vdots \\ u[N] \end{pmatrix}, \tag{4}$$

and it could be solved in principle by inverting the matrix. But this is not practical when N is very large, especially for the case in which v and u are two dimensional images and N is the number of pixels. An alternative is to apply iterative matrix inverse techniques [3].

Perhaps the simplest iterative technique is the *Jacobi* iteration. The idea is that, if we wanted to know $v[n]$ and we knew $v[m]$ for all $m \neq n$, then, using Eq. (3), the solution is easy. Jacobi iteration works in this way; i.e., at iteration $t+1$ we compute each value of $v^{t+1}[x]$ by assuming that the previous estimates, $v^t[x]$, are correct and held fixed. This produces an iteration of the form:

$$v^{t+1}[x] = \begin{cases} \frac{1}{1+2\lambda}\left(u[x]+\lambda v^t[x-1]+\lambda v^t[x+1]\right) & \text{for } 1<x<N \\ \frac{1}{1+\lambda}\left(u[1]+\lambda v^t[2]\right) & \text{for } x=1 \\ \frac{1}{1+\lambda}\left(u[N]+\lambda v^t[N-1]\right) & \text{for } x=N \end{cases}. \tag{5}$$

One can view these equations as a feedback system with linear, local feedback. Under suitable conditions the process converges (i.e., $v^{t+1}[x] \approx v^t[x]$) to a solution of Eq. 3. This occurs when

Eq. 2 is minimized. A simple condition that guarantees convergence of a Jacobi iteration is diagonal dominance of the matrix: i.e., along each row, the magnitude of the diagonal entry must be larger than the sum of magnitudes of all the off-diagonal entries. The more diagonally dominant the matrix, the faster the convergence.

Compared to other iterative methods, Jacobi iteration converges slowly. But it's the simplest to explain and to implement. One way to speed it up is to use the updated estimates, $v^{t+1}[x]$, as they become available in the update stage, rather than use the previous estimates, $v^t[x]$, in Eq. (5). This is called Gauss-Seidel iteration. For other methods and their convergence properties, see [3].

Smoothness Constraints.

The particular smoothness constraint used above is often called the (first-order) membrane model. It is a discrete approximation to the first derivative:

$$v[x+1] - v[x] \approx \frac{dv}{dx} \ .$$

This smoothness constraint is satisfied perfectly when v is a constant signal.

Another common choice for the smoothness constraint is called the (second-order) thin-plate model which is a discrete approximation to the second derivative:

$$v[x+1] - 2v[x] + v[x-1] \approx \frac{d^2v}{dx^2} \ .$$

This smoothness constraint is satisfied perfectly when v is a linear ramp signal.

But these smoothness constraints have been chosen somewhat arbitrarily. The choice of smoothness constraint is typically ad hoc, and depends on the desired smoothness of the regularized solution.

Linear Filter Interpretation.

Another interesting property of this regularization method is evident from the form of the matrix in Eq. (4). If one neglects the boundary conditions (or applies periodic boundary conditions), then this is a Toeplitz matrix, and therefore it corresponds to a linear shift-invariant operator. With the membrane smoothness constraint considered there, the impulse response was $\vec{g} = [-1, 2, -1]$. If the size of the signal (or image) is much larger than the support of the filter, as it is in this case, then a very good approximation to the solution is often obtained if we neglect the boundaries and rewrite Eq. 3 as a convolution:

$$v[x] \ * \ (\delta[x] + \lambda g[x]) \ = \ u[x] \ . \tag{6}$$

Note that the roles of the input and output signals have been reversed from our initial discussion of convolution filtering. In particular, here the output is $v[x]$ convolved with the filter kernel $\delta[x] + \lambda g[x]$, to provide the input. Recall that Eq. 3 was derived from the (first-order) membrane smoothness constraint. Starting with the (second-order) thin-plate smoothness constraint, we would get an equation of the same form but with a different filter: $\vec{g} = [1, -4, 6, -4, 1]$. Indeed, any high-pass filter could, in principle, be used instead.

Now that we have shown that regularization can be viewed as a (spatially) shift-invariant linear system, as illustrated in Eq. 6, we can also examine it in the Fourier domain. Accordingly, let's

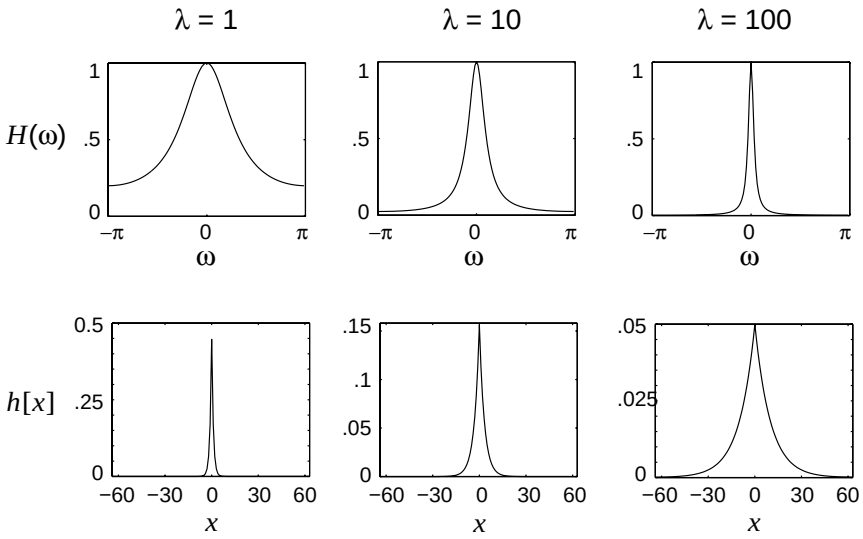

Figure 1: Frequency response $H(\omega)$ and impulse response $h[x]$ of regularization filter, for several values of λ.

take the Fourier transform (DTFT) of both sides of Eq. 6 to obtain:

$$V(\omega)\left[1+\lambda(2-e^{-i\omega}-e^{i\omega})\right] \;=\; U(\omega)\;,$$

where V and U are the Fourier transforms of v and u, respectively. This simplifies to:

$$V(\omega)\left[1+2\lambda(1-\cos(\omega))\right] \;=\; U(\omega)\;, \tag{7}$$

where we have used the identity

$$2\,\cos(\omega) = e^{-i\omega}+e^{i\omega}\;.$$

From Eq. 7 we find

$$V(\omega) \;=\; H(\omega)\,U(\omega)\;, \tag{8}$$

where

$$H(\omega) \;=\; \frac{1}{1+2\lambda(1-\cos(\omega))}\;. \tag{9}$$

Therefore, by the convolution theorem, we see that output $v[x]$ is simply the convolution of the input signal $u[x]$ with a filter kernel $h[x]$. Moreover, the Fourier transform of the filter kernel is $H(\omega)$, as given in Eq. 9.

In other words, the matrix equation (Eq. 4) is approximately equal to a convolution of the measurements $u[x]$ with an appropriate linear filter $h[x]$. If we adopt a different smoothness constraint, i.e., a different high-pass filter $g[x]$, then from Eq. 6, the equivalent feedforward linear filter has frequency response:

$$H(\omega) = \frac{1}{1+\lambda\,G(\omega)}\;, \tag{10}$$

where $G(\omega)$ is the Fourier transform of $g[x]$. Increasing the value of λ makes the output smoother, i.e., the feedforward filter becomes more lowpass (see Fig. 1).

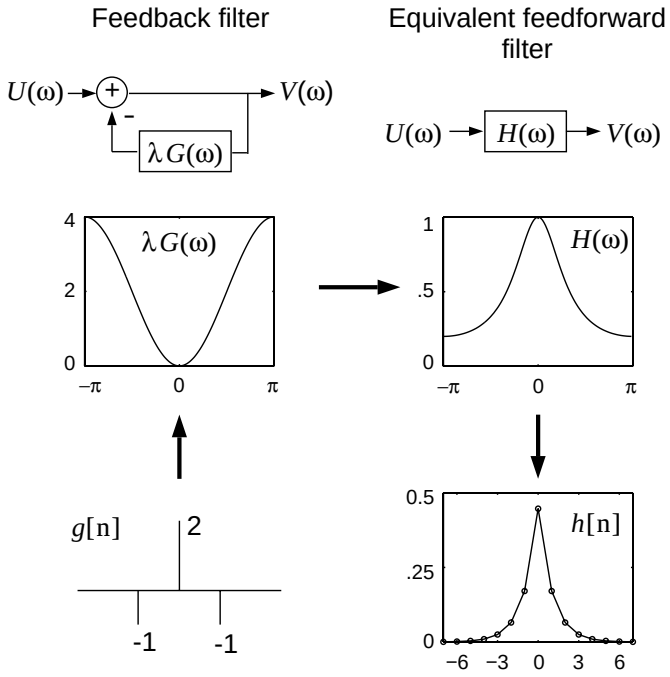

Figure 2: Regularization as a shift-invariant, feedback, linear system: $G(\omega)$ is the frequency response of the feedback filter and $H(\omega)$ is the frequency response of the equivalent feedforward filter for $\lambda = 1$

Another iterative approach is suggested by Eq. 10. For small enough values of λ such that $|\lambda G(\omega)| < 1$, we can rewrite Eq.10

$$H(\omega) = \frac{1}{1 + \lambda\, G(\omega)} = 1 - \lambda G(\omega) + (\lambda\, G(\omega))^2 - (\lambda\, G(\omega))^3 + \ldots \tag{11}$$

Since the desired output $V(\omega)$ is given by $H(\omega)U(\omega)$, we can rewrite Eq.6 as

$$v[x] = u[x] - \lambda g[x] * u[x] + \lambda g[x] * (\lambda g[x] * u[x]) - \lambda g[x] * (\lambda g[x] * (\lambda g[x] * u[x])) + \ldots \tag{12}$$

That is, $-\lambda g[x]$ is used as a recursive linear filter. The response $v[x]$ can therefore be computed as the limit (as $t \to \infty$) of the following iteration

$$\begin{aligned} v^0[x] &= u[x], \\ v^{t+1}[x] &= u[x] - \lambda g[x] * v^t[x], \quad \text{for } t \geq 0. \end{aligned}$$

A system diagram for this is given in Fig. 2.

So regularization is approximately convolution (with an appropriate filter), or recursive linear filtering (with another filter kernel).

Note that the iterative (feedback) algorithm given by Jacobi iteration in Eq. 5 (or its alternatives) has a number of practical advantages:

- First, the feedforward convolution kernel $h[x]$ is typically very large, depending on the exact choice of $g[x]$ (see Fig. 1) and the choice of λ. In particular, note that the effective spatial extent of the feedforward operator increases with λ, even though the spatial support of the

feedback computation in Eq. (5) remains unchanged. For many forms of feedback iterations, the effective support of the feedforward filter is infinite (IIR filters), in which case the feedback solution is the only practical method.

This means that we can achieve significant amounts of smoothing with only very local connectivity through feedback. But this doesn't come for free; as λ increases the matrix becomes less diagonally dominant. In this case the iterative methods take longer to converge, or even diverge. This is a trade-off between extent of spatial connectivity versus the speed of computation.

- Second, the iterative algorithm can be readily extended to deal with the general case (Eq. 1) in which data are missing (see below).

- Third, iterative methods can also be used for a wide variety of other error functions. For example, we can easily add additional terms that allow smoothness to be violated in certain places by using *robust* error norms. That is, we can replace the quadratic norm in Eq. (2) with other measures of the magnitude of errors that are more tolerant of outlying measurement errors

Interpolating Missing Data.

One of the ways that the above regularization framework generalizes naturally, is to deal with sparse measurements. For the denoising problem above, the solution was essentially a shift-invariant smoothing operator. When we have sparse data, the problem becomes one of interpolation as well.

So, consider the problem of estimating a smooth signal $v[x]$ from a set of noisy measurements u where the measurements exist at only a subset of the sample positions. Because the measurements are not available everywhere, we will have to alter the data constraint in Equation 2. The smoothness constraint, however can remain the same. This yields the following energy function that we wish to minimize:

$$E(v) = \sum_{x \in P} (v[x] - u[x])^2 + \lambda \sum_{\text{all } x} (v[x+1] - v[x])^2, \tag{13}$$

where P is a subset of sample positions where the measurements u are available. For samples where the measurements exist, the gradient constraints are the same as before (i.e., Eq. 3). For a sample position k where no measurement exists, taking the derivative of Eq. 13 with respect to $v[k]$ and setting it equal to zero gives:

$$-v[k-1] + 2\,v[k] - v[k+1] = 0.$$

The full system of linear equations can still be written as a matrix equation, i.e., it is still a linear system and it could again be solved by inverting a *big* matrix. The matrix in Eq. 4 gets replaced by a new matrix in which some of the rows look the same, but some of the rows (those corresponding to missing data) look like: $(\,0 \quad \ldots \quad -1 \quad 2 \quad -1 \quad \ldots \quad 0\,)$, with zeros substituted for the corresponding $u[k]$ on the right-hand side. Since these rows are different from the other rows, the system is no longer shift-invariant, and there is no longer a convolution kernel that could be used to compute v. The linear filter in Eq. 6 is replaced by a pair of equations:

$$0 = \begin{cases} u - v - \lambda(g * v) & \text{when } x \in P, \\ \\ (g * v) & \text{otherwise.} \end{cases} \tag{14}$$

An iterative method can again be used to solve for v. For the (first-order) membrane model smoothness constraint, the iterative equation is:

$$v^{t+1}[x] = \begin{cases} \frac{1}{1+2\lambda}\left(u[x] + \lambda v^t[x-1] + \lambda v^t[x+1]\right) & \text{for } x \in P, \\ \\ \frac{1}{2}\left(v^t[x-1] + v^t[x+1]\right) & \text{otherwise.} \end{cases} \tag{15}$$

The equations that govern the endpoints can be expressed in a similar manner.

Two-Dimensional Images.

For two-dimensional images, using the (first-order) membrane smoothness constraint, we wish to minimize:

$$\begin{aligned} E(v) &= \sum_{x,y \in P} (v[x,y] - u[x,y])^2 \\ &\quad + \lambda \sum_{\text{all } x,y} (v[x+1,y] - v[x,y])^2 + (v[x,y+1] - v[x,y])^2 \end{aligned} \tag{16}$$

where P is a subset of pixels where the measurements u are available. Taking derivatives with respect to $v[x,y]$ and setting them equal to zero gives a linear system of equations that has the same form as Eq. 14. The only difference is that the linear filter $g[x,y]$ is now 2-dimensional. For the (first-order) membrane model smoothness constraint:

$$g = \begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix}.$$

One can again solve for v iteratively. For the (first-order) membrane smoothness model, and ignoring the edge pixels (for the sake of simplicity):

$$v^{t+1}[x,y] = \begin{cases} \frac{1}{1+4\lambda}(u[x,y] + \lambda\, s^t[x,y]) & \text{when } x, y \in P, \\ \\ \frac{1}{4}\, s^t[x,y] & \text{otherwise,} \end{cases} \tag{17}$$

where $s[x,y]$ is the sum of the 4 nearest neighbors, i.e., $v[x-1,y]+v[x+1,y]+v[x,y-1]+v[x,y+1]$.

References

[1] A Blake and A Zisserman. *Visual Reconstruction.* MIT Press, Cambridge, MA, 1987.

[2] S Geman and D Geman. Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. *IEEE Pattern Analysis and Machine Intelligence*, 6:721–741, 1984.

[3] G H Golub and C F vanLoan. *Matrix Computations.* Hopkins University Press, 1983.

[4] J Marroquin, S Mitter, and T Poggio. Probabilistic solution of ill-posed problems in computational vision. *J Am Stat Assoc*, 82:76–89, 1987.

[5] T Poggio, V Torre, and C Koch. Computational vision and regularization theory. *Nature*, 317(6035):314–319, 1985.

[6] R Szeliski. Fast surface interpolation using heirarchical basis functions. *IEEE Pattern Analysis and Machine Intelligence*, 12:513–528, 1990.

[7] D Terzopoulos. Regularization of inverse visual problems involving discontinuities. *IEEE Pattern Analysis and Machine Intelligence*, 8(4):413–424, 1986.