EngSci 721

Inverse Problems and Learning From Data

Oliver Maclaren (oliver.maclaren@auckland.ac.nz)

1. *Basic concepts* [5 lectures + 1 Tutorial]

   Forward vs inverse problems. Well-posed vs ill-posed problems. Algebra and calculus of inverse problems (left and right inverses, generalised and pseudo inverses, resolution operators, matrix calculus). Representing higher dimensional problems (image data etc).

2. *Instability and regularisation in linear and nonlinear problems* [6 lectures + 1 Tutorial]

   Instability and related issues for generalised inverses. Introduction to regularisation and trade-offs. Tikhonov regularisation. Higher-order Tikhonov regularisation. Sparsity and regularisation using different norms. Truncated singular value decomposition. Iterative regularisation, including stochastic/mini-batch gradient descent.

3. *Further topics* [3 lectures + 1 Tutorial]

   Regularisation parameter choice, including statistical and machine learning views of regularisation. Confidence sets for linear and nonlinear models. Physics-informed machine learning and neural networks.

Module overview

Inverse Problems and Learning From Data (*Oliver Maclaren*)
[~14 lectures/3 tutorials]

Lecture 7: General Tikhonov Regularisation

Topics:

- Higher-order Tikhonov regularisation

Eng Sci 721 : Lecture 7

Regularisation - General Tikhonov

- Recap: basic Tikhonov
- Higher-order Tikhonov

Recap: Basic Tikhonov

The standard Tikhonov approach to constructing regularised solutions to inverse problems like

$$\boxed{\text{find } x \text{ satisfying } Ax = y}$$

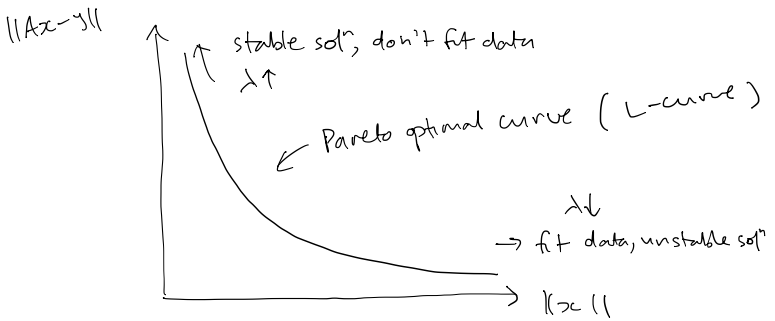
where A tends to 'smooth' or 'reduce' x, i.e. $x \to y$, is to consider the modified problem:

$$\boxed{\min_{x(\lambda)} \|Ax - y\|^2 + \lambda \|x\|^2}$$

Note that the solution depends on the regularisation parameter λ, hence we write $x(\lambda)$

Recap

— The parameter λ represents a trade-off between the importance of fitting the given data y & having a 'simple' or 'stable' solution



→ since the data y is not exact / exactly repeatable, we use λ to satisfy Hadamard's third condition of stability:

> 'small changes to given info should given small changes to solution'

→ Choosing λ is somewhat of an art & part of designing a regularisation procedure

↳ eg { L-curve corner ← we looked at.
discrepancy principle
cross-validation
etc.

Recap

— When we use the squared L_2 norm for both, it is convenient to write this as the augmented least squares problem:

$$\min \|\tilde{r}\|^2 = \min \|\tilde{A}x - \tilde{y}\|^2$$

where $\tilde{A}x = \tilde{y}$ is $\begin{bmatrix} A \\ \sqrt{\lambda} I \end{bmatrix} x - \begin{bmatrix} y \\ 0 \end{bmatrix}$

→ We can then use any standard least-squares software to solve.

Eg • Matlab: lsqminnorm
• Python: np.lstsq (does min norm if over-det).

Can also use pinv since this gives least squares sol^n. } Slightly less eff alg. I think

(Note: Matlab \ gives different pseudoinverse!)

recap

Linear vs nonlinear

The linear case has the explicit solution

$$x = (A^T A + \lambda I)^{-1} A^T y$$

ie

$$x = A_\lambda^{\#} y \quad \text{where} \quad A_\lambda^{\#} = (A^T A + \lambda I)^{-1} A^T$$

In the nonlinear case $F(x) = y$ we can still solve the 'variational form':

$$\min_{x(\lambda)} \|F(x) - y\|^2 + \lambda \|x\|^2$$

or

$$\min \|\tilde{r}\|^2 = \min \|\tilde{F}(x) - \tilde{y}\|^2$$

where is

$$\tilde{F}(x) = \tilde{y}$$

$$\begin{bmatrix} F(x) \\ \sqrt{\lambda}\, x \end{bmatrix} = \begin{bmatrix} y \\ 0 \end{bmatrix}$$

→ no explicit inverse, but...

→ solve via nonlinear least-squares, or any minimisation alg (see later)

Generalisations

The variational form is easiest to generalise

→ nonlinear problems ✓

→ different norms for data/model space ✓

Just gives an objective function to minimise via any minimisation alg.

→ scipy.minimize library (Python)

→ fminsearch (Matlab; see also opt. toolbox)

- First, however, we look at forms that still fit in least squares framework
- Then forms that fit into other efficient frameworks
  - convex opt. (eg Boyd & Vandenberghe)
- Finally (later lectures), generic problems & alternative approaches (iterative reg.)

Higher-order Tikhonov

The standard model norm that we've seen measures 'size' via

$$\|x\| \quad \text{or} \quad \|x\|^2$$

→ we can generalise this by considering

$$\|Dx\| \quad \text{or} \quad \|Dx\|^2$$

for some operator

(or $\|Dx - x_0\|$ esp. for nonlinear prob)

→ Typically D represents a first or second derivative operator, (or even a differential eqⁿ !)

↳ can consider diff. operators as 'roughening' operators (cf. integration as smoothing)

↳ we prefer smoother ie less variable solutions, hence penalise roughness.

General Tikhonov: (can be linear or nonlinear)

$$\boxed{\min \|Ax - y\|_2^2 + \lambda \|Dx\|_2^2} \quad \left.\right\} \text{ or } \lambda \|Dx - f\|_2^2 \text{ etc...}$$

where,

(for discrete, linear diff. operators in 1D → can generalise to higher D):

'zeroth order'

$$D_0 = I = \begin{bmatrix} 1 & & \\ & 1 & \\ & & \ddots \\ & & & 1 \end{bmatrix}$$

'first order'

$$D_1 = \begin{bmatrix} -1 & 1 & & & \\ & -1 & 1 & & \\ & & \ddots & \ddots & \\ & & & -1 & 1 \end{bmatrix}$$

Second order

$$D_2 = \begin{bmatrix} 1 & -2 & 1 & & \\ & 1 & -2 & 1 & \\ & & \ddots & \ddots & \ddots \\ & & & 1 & -2 & 1 \end{bmatrix}$$

Note:

$$D_1 x = \begin{bmatrix} -1 & 1 & & \\ & \ddots & \ddots & \\ & & -1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

$$= \begin{bmatrix} x_2 - x_1 \\ x_3 - x_2 \\ \vdots \\ x_n - x_{n-1} \end{bmatrix}$$

ie first order forward finite differences

$$D_2 x = \begin{bmatrix} 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

$$= \begin{bmatrix} x_1 - 2x_2 + x_3 \\ \vdots \\ x_{n-2} - 2x_{n-1} + x_n \end{bmatrix}$$

ie second-order central finite differences

Note: initial / boundary conditions?

Normally when discretising and solving differential equations, even $\frac{dy}{dx} = 0$, we need to consider boundary / initial conditions

Consider eg domain/input with three points:

$$? \quad \begin{bmatrix} -1 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

→ Can't make square

→ underdetermined!

just like $\frac{dy}{dx} = 0 \Rightarrow y = c$ for arbitrary c. } need IC/BC for det. c.

→ add rows to represent IC/BC

$\boxed{0}$ ← 'outside' grid point

eg $\boxed{-1}$ ← 'outside' part of D

$$\begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$$\equiv \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

ICs/BCs cont'd

⇒ dropping 'overflow' terms ≡ zero BCs

→ can also 'wrap' etc D 'stencil'/'kernel'

$$D_c = \begin{bmatrix} 1 & 0 & -1 \\ -1 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix}$$ ← (-1) ; } periodic/circular BCs.

→ etc!

However, when combined with data observations, these observations can play the role of BCs/ICs & so we can usually just use non-square D without ICs/BCs in inverse problems unless we want to.

$$D = \begin{bmatrix} -1 & 1 & 0 \\ 0 & -1 & 0 \end{bmatrix}$$ 2×3 OK!

If also have $y = Az$ obs. conditions.

Also:

$$D_1^2 = D_1 \cdot D_1 = \begin{bmatrix} 1 & & \\ -1 & 1 & \\ & -1 & 1 \end{bmatrix} \begin{bmatrix} 1 & & \\ -1 & 1 & \\ & -1 & 1 \end{bmatrix}$$

(with zero BC)

$$= \begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 1 & -2 & 1 \end{bmatrix}$$

$= D_2$ (with two zero BCs)

---

or 'wrapping' (i.e. 'circular differences')

$$D_{1c}^2 = D_{1c} D_{1c} = \begin{bmatrix} 1 & & -1 \\ -1 & 1 & \\ & -1 & 1 \end{bmatrix} \begin{bmatrix} 1 & & -1 \\ -1 & 1 & \\ & -1 & 1 \end{bmatrix}$$

(circular/periodic)

$$= \begin{bmatrix} 1 & 1 & -2 \\ -2 & 1 & 1 \\ 1 & -2 & 1 \end{bmatrix}$$

$= D_{2c}$

↑ 'circular'

Least squares form

Just like before, when working with the L_2-norm we can re-write

$$\min_{x(\lambda)} \|Ax - y\|_2^2 + \lambda \|Dx\|_2^2$$

as

$$\min_{x(\lambda)} \left\| \begin{bmatrix} A \\ \sqrt{\lambda} D \end{bmatrix} x - \begin{bmatrix} y \\ 0 \end{bmatrix} \right\|_2^2$$

ie

$$\min_{x(\lambda)} \|\tilde{A}x - \tilde{y}\|_2^2$$

where $\tilde{A} = \begin{bmatrix} A \\ \sqrt{\lambda} D \end{bmatrix}$

& use standard least-squares software (linear or nonlinear)

Explicit solution (linear only)

In the linear case we have the explicit solution

$$x = (A^T A + \lambda D^T D)^{-1} A^T y$$

ie

$$x = A^{\#}_{D,\lambda} y$$ (or just $x = A^{\#} y$ if lazy)

where $$A^{\#}_{D,\lambda} = (A^T A + \lambda D^T D)^{-1} A^T$$

[Note that we technically have to assume that $A^T A + \lambda D^T D$ is invertible

-> obvious for $D = I$ (see before)

-> true for $D = D_1$ or D_2 too

condition: $N(A) \cap N(D) = \{0\}$

ie null spaces have only trivial soln in common]

Examples

- Simple smoothing

$$x_{noisy} = x_{smooth} + \epsilon = I x_{smooth} + \epsilon$$

$$= A x_{smooth} + \epsilon \,, \quad \epsilon \text{ unknown}$$

$$\Rightarrow \boxed{A = I}$$ ('deterministic' part; ϵ dealt with by using least squares sol')

Note: we don't explicitly model ϵ, but relates to choice of 'fit'

Here: $\| A x_{smooth} - x_{noisy} \|_2^2$

→ least squares fit (don't expect exact fit when noisy).

→ but without reg., will fit exactly $\Big\} A^{-1} = I^{-1} = I$

Regularised:

$$\boxed{x_{smooth} = \left(I + \lambda D^T D \right)^{-1} I \, x_{noisy}}$$

solves: $\| x_{noisy} - x_{smooth} \|_2^2 + \lambda \| D x_{smooth} \|_2^2$

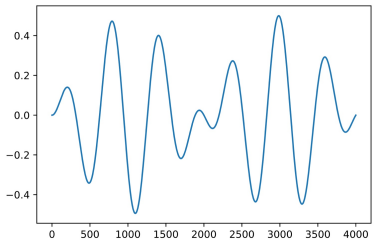
fit data — smooth data.

Example

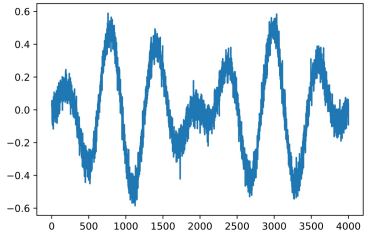
[see Boyd & Vandenberghe 6.3.3 (attached) for original]

My attempt

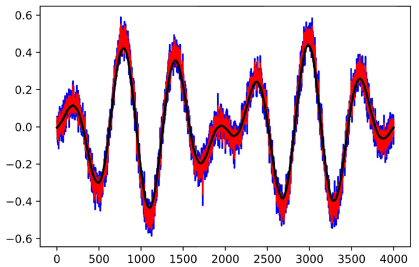
True signal



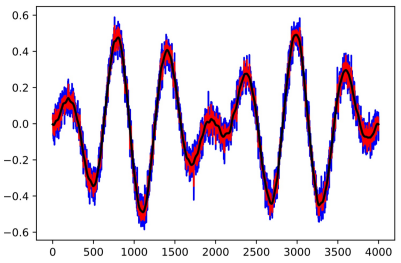
Noisy signal



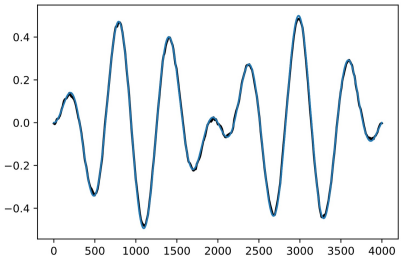
Recovered (D_1 reg.) diff. λ



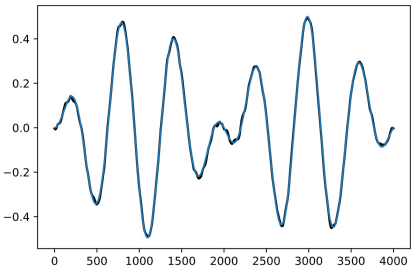
Recovered (D_2 reg.) diff. λ



'Best D_1' & True



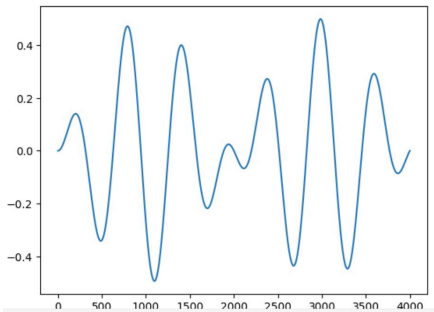
'Best D_2' & True



Code snippets

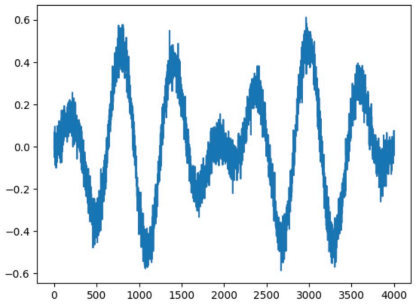
```
signal = 0.5*np.sin(2*t*np.pi/n)*np.sin(0.01*t)
plt.plot(t,signal)
plt.show()
```

✓ 0.2s



```
signal_noisy = signal + 0.05*np.random.normal(0,1,size=len(t))
plt.plot(t,signal_noisy)
plt.show()
```

✓ 0.1s



OUTPUT DEBUG CONSOLE TERMINAL PORTS JUPYTER

Forward model

```
A = np.eye(len(signal_noisy))
print(A)
```

✓ 0.0s

```
[[1. 0. 0. ... 0. 0. 0.]
 [0. 1. 0. ... 0. 0. 0.]
 [0. 0. 1. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 1. 0. 0.]
 [0. 0. 0. ... 0. 1. 0.]
 [0. 0. 0. ... 0. 0. 1.]]
```

First derviative matrix for true signal (model)

```
D1 = np.zeros((A.shape[0]-1,A.shape[1]))
diff_row = np.zeros(len(signal_noisy))
diff_row[0] = -1
diff_row[1] = 1
for i in range(0,D1.shape[0]):
    D1[i,:] = np.roll(diff_row,i)
print(D1)
```

✓ 0.0s

```
[[-1.  1.  0. ...  0.  0.  0.]
 [ 0. -1.  1. ...  0.  0.  0.]
 [ 0.  0. -1. ...  0.  0.  0.]
 ...
 [ 0.  0.  0. ...  1.  0.  0.]
 [ 0.  0.  0. ... -1.  1.  0.]
 [ 0.  0.  0. ...  0. -1.  1.]]
```

Second derviative (model)

```
D2 = np.zeros((A.shape[0]-2,A.shape[1]))
second_diff_row = np.zeros(len(signal_noisy))
second_diff_row [0] = 1
second_diff_row [1] = -2
second_diff_row [2] = 1
for i in range(0,D2.shape[0]):
    D2[i,:] = np.roll(second_diff_row ,i)
print(D2)
```

✓ 0.0s

```
[[ 1. -2.  1. ...  0.  0.  0.]
 [ 0.  1. -2. ...  0.  0.  0.]
 [ 0.  0.  1. ...  0.  0.  0.]
 ...
 [ 0.  0.  0. ...  1.  0.  0.]
 [ 0.  0.  0. ... -2.  1.  0.]
 [ 0.  0.  0. ...  1. -2.  1.]]
```

Code snippets : results

Inversions

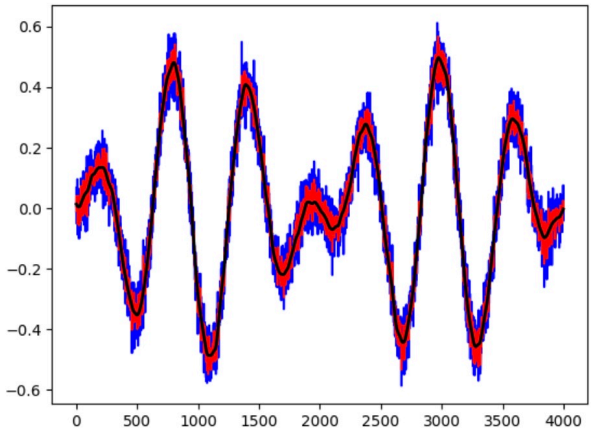
Use sparse matrices and lsqr due to size and sparsity of matrices. Speeds up a lot!

```
from scipy.sparse import csc_matrix, coo_array
from scipy.sparse.linalg import lsqr
alphav = np.logspace(-2, 2.0, num=10)
#augment as function of regularisation parameter alpha and D
D = np.copy(D2)
Ap_aug = lambda alpha, D : np.vstack([A,alpha*D])
signal_smooth_solns = np.zeros((len(signal_noisy),len(alphav)))
signal_noisy_aug = np.hstack([signal_noisy,np.zeros(D.shape[0])])
for i, alpha in enumerate(alphav):
    # solve as sparse linear (overdetermined) system with least squares
    signal_smooth_solns[:,i] = lsqr(coo_array(Ap_aug(alpha,D)), signal_noisy_aug)[0]
```

✓ 4.8s

```
plt.plot(signal_smooth_solns[:,0],'b')
plt.plot(signal_smooth_solns[:,4],'r')
plt.plot(signal_smooth_solns[:,9],'k',linewidth=2)
plt.show()
```

✓ 0.1s



'Physics-informed machine learning'?

As the regularisation term is made more sophisticated it can even represent eg an actual 'scientific' mathematical model like a differential eqn or system of differential equations.

General idea is eg

$$\|y - f(x)\|^2 + \lambda \|M(x)\|^2$$

data fit, $f(x)$ a flexible stat/ML model like neural networks, spline, Gaussian processes etc.

fit to scientific model, model written as $M(x)=0$ eg differential eqn.

→ The scientific model is not exactly satisfied, rather approximately satisfied

→ Called:
- physics-informed ML
- functional/differential data analysis
- generalised profiling.

→ see assignment + later!

Boyd & Vandenberghe (Convex Optimization)

Quadratic smoothing

The simplest reconstruction method uses the quadratic smoothing function

$$\phi_{\rm quad}(x) = \sum_{i=1}^{n-1}(x_{i+1} - x_i)^2 = \|Dx\|_2^2,$$

where $D \in \mathbf{R}^{(n-1)\times n}$ is the bidiagonal matrix

$$D = \begin{bmatrix} -1 & 1 & 0 & \cdots & 0 & 0 & 0 \\ 0 & -1 & 1 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & -1 & 1 & 0 \\ 0 & 0 & 0 & \cdots & 0 & -1 & 1 \end{bmatrix}.$$

We can obtain the optimal trade-off between $\|\hat{x} - x_{\rm cor}\|_2$ and $\|D\hat{x}\|_2$ by minimizing

$$\|\hat{x} - x_{\rm cor}\|_2^2 + \delta\|D\hat{x}\|_2^2,$$

where $\delta > 0$ parametrizes the optimal trade-off curve. The solution of this quadratic problem,

$$\hat{x} = (I + \delta D^T D)^{-1} x_{\rm cor},$$

can be computed very efficiently since $I + \delta D^T D$ is tridiagonal; see appendix C.

Quadratic smoothing example

Figure 6.8 shows a signal $x \in \mathbf{R}^{4000}$ (top) and the corrupted signal $x_{\rm cor}$ (bottom). The optimal trade-off curve between the objectives $\|\hat{x} - x_{\rm cor}\|_2$ and $\|D\hat{x}\|_2$ is shown in figure 6.9. The extreme point on the left of the trade-off curve corresponds to $\hat{x} = x_{\rm cor}$, and has objective value $\|Dx_{\rm cor}\|_2 = 4.4$. The extreme point on the right corresponds to $\hat{x} = 0$, for which $\|\hat{x} - x_{\rm cor}\|_2 = \|x_{\rm cor}\|_2 = 16.2$. Note the clear knee in the trade-off curve near $\|\hat{x} - x_{\rm cor}\|_2 \approx 3$.

Figure 6.10 shows three smoothed signals on the optimal trade-off curve, corresponding to $\|\hat{x} - x_{\rm cor}\|_2 = 8$ (top), 3 (middle), and 1 (bottom). Comparing the reconstructed signals with the original signal x, we see that the best reconstruction is obtained for $\|\hat{x} - x_{\rm cor}\|_2 = 3$, which corresponds to the knee of the trade-off curve. For higher values of $\|\hat{x} - x_{\rm cor}\|_2$, there is too much smoothing; for smaller values there is too little smoothing.

Total variation reconstruction

Simple quadratic smoothing works well as a reconstruction method when the original signal is very smooth, and the noise is rapidly varying. But any rapid variations in the original signal will, obviously, be attenuated or removed by quadratic smoothing. In this section we describe a reconstruction method that can remove much of the noise, while still preserving occasional rapid variations in the original signal. The method is based on the smoothing function

$$\phi_{\rm tv}(\hat{x}) = \sum_{i=1}^{n-1}|\hat{x}_{i+1} - \hat{x}_i| = \|D\hat{x}\|_1,$$

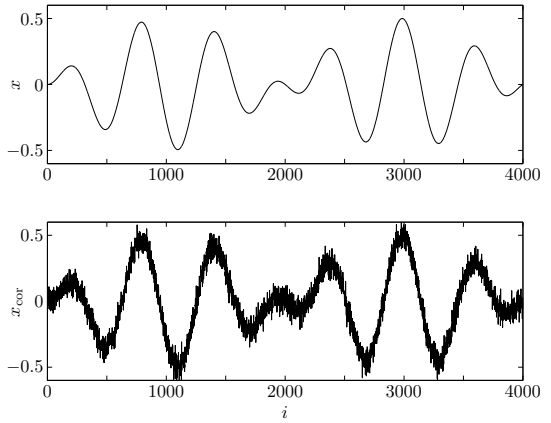


Figure 6.8 *Top:* the original signal $x \in \mathbf{R}^{4000}$. *Bottom:* the corrupted signal $x_{\rm cor}$.

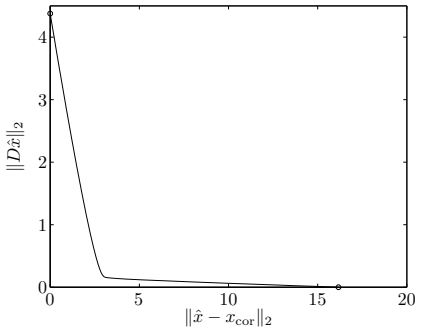


Figure 6.9 Optimal trade-off curve between $\|D\hat{x}\|_2$ and $\|\hat{x} - x_{\rm cor}\|_2$. The curve has a clear knee near $\|\hat{x} - x_{\rm cor}\| \approx 3$.

Boyd & Vandenberghe (Convex Optimization)

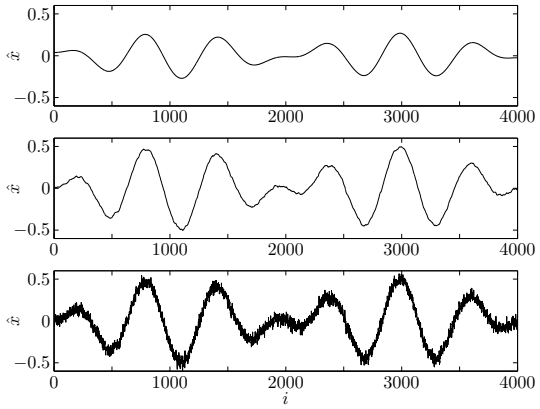


Figure 6.10 Three smoothed or reconstructed signals $\hat{x}$. The top one corresponds to $\|\hat{x} - x_{\text{cor}}\|_2 = 8$, the middle one to $\|\hat{x} - x_{\text{cor}}\|_2 = 3$, and the bottom one to $\|\hat{x} - x_{\text{cor}}\|_2 = 1$.

which is called the *total variation* of $x \in \mathbf{R}^n$. Like the quadratic smoothness measure ϕ_{quad}, the total variation function assigns large values to rapidly varying $\hat{x}$. The total variation measure, however, assigns relatively less penalty to large values of $|x_{i+1} - x_i|$.

Total variation reconstruction example

Figure 6.11 shows a signal $x \in \mathbf{R}^{2000}$ (in the top plot), and the signal corrupted with noise x_{cor}. The signal is mostly smooth, but has several rapid variations or jumps in value; the noise is rapidly varying.

We first use quadratic smoothing. Figure 6.12 shows three smoothed signals on the optimal trade-off curve between $\|D\hat{x}\|_2$ and $\|\hat{x} - x_{\text{cor}}\|_2$. In the first two signals, the rapid variations in the original signal are also smoothed. In the third signal the steep edges in the signal are better preserved, but there is still a significant amount of noise left.

Now we demonstrate total variation reconstruction. Figure 6.13 shows the optimal trade-off curve between $\|D\hat{x}\|_1$ and $\|\hat{x} - x_{\text{corr}}\|_2$. Figure 6.14 shows the reconstructed signals on the optimal trade-off curve, for $\|D\hat{x}\|_1 = 5$ (top), $\|D\hat{x}\|_1 = 8$ (middle), and $\|D\hat{x}\|_1 = 10$ (bottom). We observe that, unlike quadratic smoothing, total variation reconstruction preserves the sharp transitions in the signal.

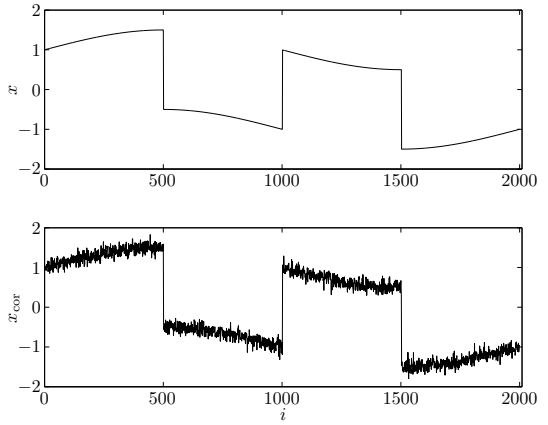


Figure 6.11 A signal $x \in \mathbf{R}^{2000}$, and the corrupted signal $x_{\text{cor}} \in \mathbf{R}^{2000}$. The noise is rapidly varying, and the signal is mostly smooth, with a few rapid variations.