Math 6397 Bayesian Inverse Problems

Problem Set 2

Due on Friday, April 7, at 10:00 PM

1 Background

In the 2D setting the model equation is given by

$$y(s_1, s_2) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \mathrm{ker}(s_1 - t_1, s_2 - t_2)x(t_1, t_2)\,\mathrm{d}t_1\,\mathrm{d}t_2, \quad 0 \leq t_1, t_2 \leq 1. \tag{1}$$

The kernel $\mathrm{ker} : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$, $\mathrm{ker} := \mathrm{ker}_\tau$, is separable and given by

$$\mathrm{ker}_\tau(s_1, s_2) = \frac{1}{\sqrt{(2\pi)^2\tau_1^2\tau_2^2}} \exp\left(-\frac{1}{2}\left(\frac{s_1^2}{\tau_1^2} + \frac{s_2^2}{\tau_2^2}\right)\right) = \mathrm{ker}_{\tau_1}(s_1)\,\mathrm{ker}_{\tau_2}(s_2),$$

where

$$\mathrm{ker}_{\tau_i}(s_i) = \frac{1}{\sqrt{2\pi\tau_i^2}} \exp\left(-\frac{s_i^2}{2\tau_i^2}\right)$$

with $\tau := (\tau_1, \tau_2)$, $\tau_i > 0$. Under the assumption that $\mathrm{ker}_\tau(s_1, s_2) = 0$ for $(s_1, s_2) \notin (-1, 1)^2$, the integration again reduces from the interval $(-\infty, \infty)$ to the interval $(-1, 2)$. The equation above represents a Fredholm first kind integral equation. See Fig. 1 for an illustration.

The *direct problem* associated with (1) is the following: Given the source function x and the kernel ker determine the blurred data y. The *inverse problem* associated with (1) is as follows: Given the kernel ker and the blurred data y determine the source x. We will refer to this problem as *source reconstruction problem*. Likewise, we can consider the problem of reconstructing the kernel ker from data y. We will refer to this problem as a *kernel reconstruction problem*.

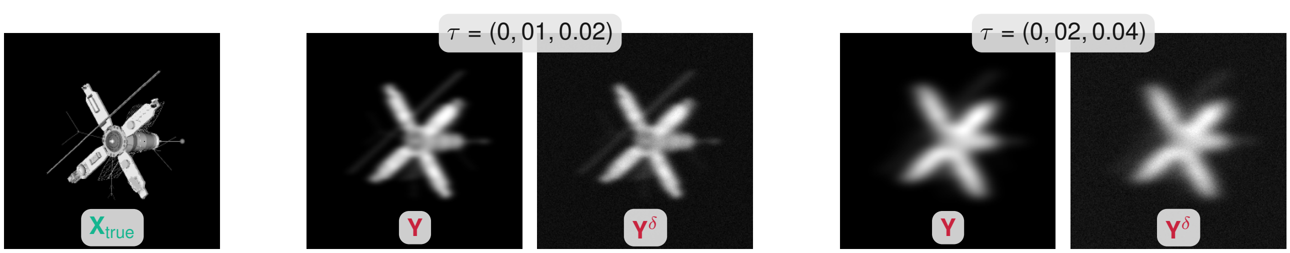

Figure 1: Two-dimensional deconvolution problem. The problem dimensions are 256×256. The leftmost plot shows the dataset $\mathbf{X}_{\text{true}}$. The blocks in the middle and on the right show the perturbed data $\mathbf{Y}$ and $\mathbf{Y}^\delta$ for different values of $\tau = (\tau_1, \tau_2)$.

1.1 Numerical Discretization

Likewise to the 1D case, we consider a midpoint quadrature for discretization with uniform mesh size $h = 1/n$ and coordinates

$$(t_{1,i_1}, t_{2,i_2}) = ((i_1 - \frac{1}{2})h, (i_2 - \frac{1}{2})h), \quad i_1, i_2 = -n+1, \ldots, 2n,$$

and $(s_{1,j_1}, s_{2,j_2}) = (t_{1,j_1}, t_{2,j_2})$ for $j_1, j_2 = 1, \ldots, n$. Moreover, let $y_{j_1 j_2} = y(s_{1,j_1}, s_{2,j_2})$ and $x_{i_1 i_2} = x(t_{1,i_1}, t_{2,i_2})$, and $\ker(s_{1,j_1} - t_{1,l_1}, s_{2,j_2} - t_{2,l_2}) = \ker((j_1 - i_1)h, (j_2 - i_2)h)$ with $\ker(r_1 h, r_2 h) =: \kappa_{r_1,r_2} \neq 0$ only for $= n+1 \leq r_1, r_2 \leq n-1$. Under these assumptions we can approximate the above equation by the system of linear equations

$$y_{j_1 j_2} = h^2 \sum_{i_1 = j_1 - n+1}^{j_1+n-1} \sum_{i_2 = j_2 - n+1}^{j_2+n-1} \kappa_{j_1 - i_1, j_2 - i_2} x_{i_1 i_2}, \quad j_1, j_2 = 1, \ldots, n.$$

Since the kernel is separable, we have

$$\kappa_{j_1 - i_1, j_2 - i_2} = \ker_{\tau}((j_1 - i_1)h_1, (j_2 - i_2)h_2) = \ker_{\tau_1}((j_1 - i_1)h_1) \ker_{\tau_2}((j_2 - i_2)h_2).$$

With $\kappa_{1,j_1-i_1} := \ker_{\tau_1}((j_1 - i_1)h_1)$ and $\kappa_{2,j_2-i_2} := \ker_{\tau_2}((j_2 - i_2)h_2)$, we obtain

$$y_{j_1 j_2} = h \sum_{i_2=1}^{n} \kappa_{2,j_2-i_2} \left(h \sum_{i_1=1}^{n} \kappa_{1,j_1-i_1} x_{i_1 i_2} \right), \quad j_1, j_2 = 1, \ldots, n. \tag{2}$$

For a compact representation, we assemble $x_{i_1 i_2}$ and $y_{j_1 j_2}$ as $n \times n$ matrices

$$\mathbf{Y} := \left[y_{j_1 j_2} \right]_{j_1, j_2 = 1}^{n,n}, \quad \text{and} \quad \mathbf{X} := \left[x_{i_1 i_2} \right]_{i_1, i_2 = 1}^{n,n},$$

respectively. Moreover, we construct 1D kernel matrices $\mathbf{K}_1$ and $\mathbf{K}_2$ for each spatial direction. More precisely, we have

$$\mathbf{K}_1 = \left[k_{1,i_1 j_1} \right]_{i_1, j_1 = 1}^{n,n} \quad \text{and} \quad \mathbf{K}_2 = \left[k_{2,i_2 j_2} \right]_{i_2, j_2 = 1}^{n,n},$$

with entries $k_{1,i_1 j_1} = h\kappa_{1,j_1-i_1} = h \ker_{\tau_1}((j_1 - i_1)h)$ and $k_{2,i_2 j_2} = h\kappa_{2,j_2-i_2} = h \ker_{\tau_2}((j_2 - i_2)h)$, respectively. With this, (2) can be expressed as

$$\begin{aligned} y_{j_1 j_2} &= h \sum_{i_2=1}^{n} \kappa_{2,j_2-i_2} \left(h \sum_{i_1=1}^{n} \kappa_{1,j_1-i_1} x_{i_1 i_2} \right) = \sum_{i_2=1}^{n} \kappa_{2,j_2-i_2} \left[\mathbf{K}_1 \mathbf{X} \right]_{j_1, i_2} \\ &= \sum_{i_2=1}^{n} \kappa_{2,j_2-i_2} \left[(\mathbf{K}_1 \mathbf{X})^\mathsf{T} \right]_{i_2, j_1} = \left[\mathbf{K}_2 (\mathbf{K}_1 \mathbf{X})^\mathsf{T} \right]_{j_2, j_1} \end{aligned},$$

and by that

$$\mathbf{Y} = (\mathbf{K}_2 (\mathbf{K}_1 \mathbf{X})^\mathsf{T})^\mathsf{T} = \mathbf{K}_1 \mathbf{X} \mathbf{K}_2^\mathsf{T}. \tag{3}$$

The representation in (3) will be useful in computation. In fact, forming the dense convolution operator $\mathbf{K}$ for the 2D case can become prohibitively expensive. To see this, let $\text{vec} : \mathbb{R}^{m,n} \to \mathbb{R}^{nm}$ be an operator that maps an $n \times m$ matrix into lexicographical ordering. We obtain

$$\mathbf{K}\mathbf{x} = \mathbf{y} \tag{4}$$

with $\mathbf{y} = \text{vec}(\mathbf{Y}) \in \mathbb{R}^{nn}$, $\mathbf{x} = \text{vec}(\mathbf{X}) \in \mathbb{R}^{nn}$, and $\mathbf{K}\mathbf{x} = \text{vec}(\mathbf{K}_1 \mathbf{X} \mathbf{K}_2^\mathsf{T}) = (\mathbf{K}_2 \otimes \mathbf{K}_1)\mathbf{x}$ with $\mathbf{K}_2 \otimes \mathbf{K}_1 \in \mathbb{R}^{nn,nn}$. Here, $\otimes$ denotes the Kronecker product. An implementation for computing the kernel matrices $\mathbf{K}_1$ and $\mathbf{K}_2$ for a given mesh size n (and parameters $\tau_1, \tau_2 > 0$) for the two-dimensional case can be found in prbsets/deconv2D/getKernel2D.m.

1.2 Efficient Computations Using the SVD

We can represent $\mathbf{K}$ based on the SVD of $\mathbf{K}_1$ and $\mathbf{K}_2$. In particular, if $\mathbf{K}_1 = \mathbf{U}_1\mathbf{S}_1\mathbf{V}_1$ and $\mathbf{K}_2 = \mathbf{U}_2\mathbf{S}_2\mathbf{V}_2$, then

$$\mathbf{K} = (\mathbf{U}_2 \otimes \mathbf{U}_1)(\mathbf{S}_2 \otimes \mathbf{S}_1)(\mathbf{V}_2^\mathsf{T} \otimes \mathbf{V}_1^\mathsf{T}). \tag{5}$$

(Notice that the diagonal entries of $\mathbf{S}_2 \otimes \mathbf{S}_1$ are no longer in descending order.) Consequently, the least squares estimator can be expressed as

$$\mathbf{x}_{\mathrm{LS}} = \mathbf{K}^\dagger \mathbf{y} = (\mathbf{V}_2 \otimes \mathbf{V}_1)(\mathbf{S}_2^\dagger \otimes \mathbf{S}_1^\dagger)(\mathbf{U}_2^\mathsf{T} \otimes \mathbf{U}_1^\mathsf{T})\mathbf{y}. \tag{6}$$

where $\mathbf{A}^\dagger$ denotes the pseudoinverse (or generalized inverse) of $\mathbf{A}$. Equivalently, we have

$$\mathbf{X}_{\mathrm{LS}} = \mathbf{K}_1^\dagger \mathbf{Y}(\mathbf{K}_2^\mathsf{T})^\dagger = \mathbf{V}_1(\mathbf{S}_1^\dagger(\mathbf{U}_1^\mathsf{T}\mathbf{Y}\mathbf{U}_2)(\mathbf{S}_2^\dagger)^\mathsf{T})\mathbf{V}_2^\mathsf{T}. \tag{7}$$

Moreover, it can be shown that we can represent the matrix vector product $\mathbf{A}\mathbf{x}$ as

$$\mathbf{A}\mathbf{x} = \mathrm{vec}(\mathbf{U}_1((\mathbf{s}_1\mathbf{s}_2^\mathsf{T}) \odot (\mathbf{V}_1^\mathsf{T}\mathbf{X}\mathbf{V}_2))\mathbf{U}_2^\mathsf{T}), \tag{8}$$

with $\mathbf{S}_1 = \mathrm{diag}(\mathbf{s}_1)$ and with $\mathbf{S}_2 = \mathrm{diag}(\mathbf{s}_2)$, respectively; $\odot$ denotes the Hadamard product (i.e., an entrywise matrix-matrix product). Using similar arguments, it can be shown that we can represent the Tikhonov solution using an SVD of $\mathbf{K}_1$ and $\mathbf{K}_2$ as

$$\mathbf{X}_\alpha = \mathbf{V}_1 \left(\left((\mathbf{s}_1\mathbf{s}_2^\mathsf{T}) \oslash ((\mathbf{s}_1\mathbf{s}_2^\mathsf{T})^{\circ 2} + \alpha \mathbf{E} \right) \odot (\mathbf{U}_1^\mathsf{T}\mathbf{Y}\mathbf{U}_2) \right) \mathbf{V}_2^\mathsf{T}, \tag{9}$$

where $\circ 2$ denotes the elementwise square (Hadamard power), i.e., $(\mathbf{s}_1\mathbf{s}_2^\mathsf{T})^{\circ 2} = (\mathbf{s}_1\mathbf{s}_2^\mathsf{T}) \odot (\mathbf{s}_1\mathbf{s}_2^\mathsf{T})$, $\oslash$ is the Hadamard division (i.e., elementwise matrix-matrix division), and $\mathbf{E}$ is an $n \times n$ matrix of all ones.

1.3 Additive Noise Model

To compute the observation $\mathbf{y} := \mathbf{y}^\delta \in \mathbb{R}^n$, we apply $\mathbf{K} \in \mathbb{R}^{n,n}$ to $\mathbf{x}_{\mathrm{true}} \in \mathbb{R}^n$ and perturb the resulting $\mathbf{y} \in \mathbb{R}^n$ by noise $\delta\boldsymbol{\eta} \in \mathbb{R}^n$, i.e., $\mathbf{y}^\delta = \mathbf{K}\mathbf{x}_{\mathrm{true}} + \delta\boldsymbol{\eta}$. The noise level δ will be selected such that the signal-to-noise ratio $\|\mathbf{K}\mathbf{x}_{\mathrm{true}}\|/\sqrt{n\delta^2}$ is equal to a constant γ. An implementation of this additive noise model can be found in core/addNoise.m.

2 Assignments

1. Suppose we form and store the full matrix $\mathbf{K}$ (similar to the one-dimensional example) for a two dimensional source $\mathbf{X}$ of size 128×128, 256×256, and 512×512. How much memory would this require, assuming that we store $\mathbf{K}$ in double precision (i.e., numeric values will occupy 64 bit = 8 byte in computer memory)?

2. Let $\mathbf{A} \in \mathbb{R}^{m,n}$, $\mathbf{B} \in \mathbb{R}^{r,s}$. Then, the Kronecker product is given by

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & a_{12}\mathbf{B} & \cdots & a_{1n}\mathbf{B} \\ a_{21}\mathbf{B} & a_{22}\mathbf{B} & \cdots & a_{2n}\mathbf{B} \\ \vdots & \vdots & & \vdots \\ a_{m1}\mathbf{B} & a_{m2}\mathbf{B} & \cdots & a_{mn}\mathbf{B} \end{bmatrix} \in \mathbb{R}^{mr,ns}.$$

a) Use the identities $(\mathbf{A} \otimes \mathbf{B})^\mathsf{T} = \mathbf{A}^\mathsf{T} \otimes \mathbf{B}^\mathsf{T}$, $(\mathbf{A} \otimes \mathbf{B})^\dagger = \mathbf{A}^\dagger \otimes \mathbf{B}^\dagger$ and $(\mathbf{A} \otimes \mathbf{B})(\mathbf{C} \otimes \mathbf{D}) = (\mathbf{AC} \otimes \mathbf{BD})$, where $\mathbf{C} \in \mathbb{R}^{n,p}$ and $\mathbf{D} \in \mathbb{R}^{s,t}$, to proof (5) and (6), respectively.

b) Let $\mathbf{A} \in \mathbb{R}^{m,n}$, $\mathbf{B} \in \mathbb{R}^{r,s}$, $\mathbf{C} \in \mathbb{R}^{s,n}$. Use the identity $\mathrm{vec}(\mathbf{BCA}^\mathsf{T}) = \mathbf{A} \otimes \mathbf{B}\,\mathrm{vec}(\mathbf{C})$ to show that (6) and (7) are equivalent, i.e.,

$$\mathrm{vec}(\mathbf{V}_1(\mathbf{S}_1^\dagger(\mathbf{U}_1^\mathsf{T}\mathbf{Y}\mathbf{U}_2)(\mathbf{S}_2^\dagger)^\mathsf{T})\mathbf{V}_2^\mathsf{T}) = (\mathbf{V}_2 \otimes \mathbf{V}_1)(\mathbf{S}_2^\dagger \otimes \mathbf{S}_1^\dagger)(\mathbf{U}_2^\mathsf{T} \otimes \mathbf{U}_1^\mathsf{T})\mathbf{y}.$$

(Notice that this identity is a generalization of $\mathbf{Kx} = \mathrm{vec}(\mathbf{K}_1\mathbf{X}\mathbf{K}_2^\mathsf{T}) = (\mathbf{K}_2 \otimes \mathbf{K}_1)\mathbf{x}$.)

c) Let $\mathbf{A} \in \mathbb{R}^{m,n}$, $\mathbf{B} \in \mathbb{R}^{r,s}$, $\mathbf{C} \in \mathbb{R}^{s,n}$, $\mathbf{a} \in \mathbb{R}^m$ and $\mathbf{b} \in \mathbb{R}^n$. Use the identities $\mathrm{vec}(\mathbf{BCA}^\mathsf{T}) = \mathbf{A} \otimes \mathbf{B}\,\mathrm{vec}(\mathbf{C})$ and $\mathrm{diag}(\mathbf{a}) \otimes \mathrm{diag}(\mathbf{b}) = \mathrm{diag}(\mathrm{vec}(\mathbf{ba}^\mathsf{T}))$ to show that (8) holds.

d) Proof that the Tikhonov regularized solution can be expressed as (9).

3. Next, we consider computing a solution to the two-dimensional problem of the form (1) using direct methods. We will exploit the fact that $\mathbf{K}$ is separable.

   a) According to (5) we can represent the SVD of $\mathbf{K}$ in terms of the SVDs of $\mathbf{K}_1$ and $\mathbf{K}_2$, respectively. It can be shown that the right-singular vectors of $\mathbf{K}_2 \otimes \mathbf{K}_1$ (i.e., the columns of $\mathbf{V}_2 \otimes \mathbf{V}_1$) can be represented as $\mathrm{vec}(\mathbf{v}_{1,i}\mathbf{v}_{2,j}^\mathsf{T})$, where $\mathbf{v}_{1,i}$ and $\mathbf{v}_{2,j}$ are the ith and jth column of $\mathbf{V}_1$ and $\mathbf{V}_2$, respectively. Visualize the outer product $\mathbf{s}_1\mathbf{s}_2^\mathsf{T}$ in logarithmic scale using Matlab's `imagesc` command. Moreover, visualize the right singular vectors $\mathbf{v}_{1,i}\mathbf{v}_{2,j}^\mathsf{T}$ for all possible pairs of $i, j = 1, 4, 16$ using Matlab's `imagesc` command. What do you notice for increasing i, j about the sungalar value/vector pairs $\{(\mathbf{s}_1\mathbf{s}_2^\mathsf{T})_{ij}, \mathbf{v}_{1,i}\mathbf{v}_{2,j}^\mathsf{T}\}$? **Hint:** *A template for your implementation is prbsets/deconv2D/scDeconvSVD2D.m.*

   b) Solve the inverse problem using a direct method. In particular, compute the Tikhonov solution $\mathbf{X}_\alpha$ based on (9). Compare your solution to the least squares solution $\mathbf{X}_{\mathrm{ls}}$ in (7). **Hint:** *For the least squares solution, use $\mathbf{K}_1^\dagger\mathbf{Y}(\mathbf{K}_2^\mathsf{T})^\dagger$. You can use Matlab's forward and backward slash operator to compute/apply the generalized inverses. A template for your implementation is prbsets/deconv2D/scDeconvTRegDirSVD2D.m.*

4. Next, we consider an iterative method to solve the optimality conditions

$$\mathbf{K}^\mathsf{T}(\mathbf{K}\mathbf{x}^\star - \mathbf{y}^\delta) + \alpha\mathbf{x}^\star = \mathbf{0} \tag{10}$$

   of the Tikhonov-regularized problem (as opposed to a direct method). This allows us to avoid explicitly forming and/or storing the matrix operator $\mathbf{K} \in \mathbb{R}^{n,n}$. In particular, we will consider a matrix-free (preconditioned) conjugate gradient (**CG**) method to solve the linear system (10) for $\mathbf{x}^\star$. This Krylov subspace method only requires an expression for the action of a matrix on a vector (i.e., an expression for the matrix-vector-product ("matvec")). In exact arithmetic it is guaranteed that the CG converges to a solution after at most r iterations, where r is the number of distinct eigenvalues of the matrix of the linear system.

   a) Implement a CG algorithm. **Hint:** *A template for implementing the CG algorithm is core/runCG.m. A script to test your CG code is xmpl/exSolLSCG.m.*

b) Use your CG algorithm to solve (10) for $\mathbf{x}^\star$. To evaluate this matrix vector product, we will apply one-dimensional blurring operators $\mathbf{K}_1$ and $\mathbf{K}_2$ along the individual coordinate directions. This results in significant savings in terms of memory-requirements. This expression for the matvec is given in (3). Notice that you need to apply $\mathbf{K}$ and $\mathbf{K}^\mathsf{T}$ (see (10)). **Hint:** *One complication is that CG expects the data in a column vector (lexicographical ordering; $nn \times 1$ column vector $\mathbf{x}$) whereas the matvec is defined for an $n \times n$ matrix $\mathbf{X}$. You can use Matlab's `reshape` to move between layouts. You can use Matlab's `pcg` implementation to set up everything before using your own implementation. A template for your implementation is prbsets/deconv2D/scDeconvTRegCGMF2D.m.*