EE263 Homework 4

Summer 2022

3.300. Orthogonal complement of a subspace. If $\mathcal{V}$ is a subspace of $\mathbb{R}^n$ we define $\mathcal{V}^\perp$ as the set of vectors orthogonal to every element in $\mathcal{V}$, *i.e.*,

$$\mathcal{V}^\perp = \{\ x \mid \langle x, y\rangle = 0,\ \forall y \in \mathcal{V}\ \}.$$

a) Verify that $\mathcal{V}^\perp$ is a subspace of $\mathbb{R}^n$.

b) Suppose $\mathcal{V}$ is described as the span of some vectors $v_1, v_2, \ldots, v_r$. Express $\mathcal{V}$ and $\mathcal{V}^\perp$ in terms of the matrix $V = \begin{bmatrix} v_1 & v_2 & \cdots & v_r \end{bmatrix} \in \mathbb{R}^{n\times r}$ using common terms (range, nullspace, transpose, etc.)

c) Show that every $x \in \mathbb{R}^n$ can be expressed uniquely as $x = v + v^\perp$ where $v \in \mathcal{V}$, $v^\perp \in \mathcal{V}^\perp$. *Hint:* let v be the projection of x on $\mathcal{V}$.

d) Show that $\dim \mathcal{V}^\perp + \dim \mathcal{V} = n$.

e) Show that $\mathcal{V} \subseteq \mathcal{U}$ implies $\mathcal{U}^\perp \subseteq \mathcal{V}^\perp$.

6.190. Using data to choose the basis functions. In this question, we are given data, consisting of (s^i, g^i) pairs, for $i = 1, \ldots, m$, with $s^i \in \mathbb{R}^d$, and $g^i \in \mathbb{R}$. We would like to approximately fit this data with a linear combination of functions $f_1, \ldots, f_n : \mathbb{R}^d \to \mathbb{R}$ function so that

$$g_i \approx x_1 f_1(s^i) + \cdots + x_n f_n(s^i)$$

We will use this approximation to predict the function at other points $q \in \mathbb{R}^d$, using predictor

$$\hat{g}(q) = x_1 f_1(q) + \cdots + x_n f_n(q)$$

In class we have seen how the basis functions or regressors f_i may be chosen as polynomials. However, instead in this question you will develop a different approach, where the basis functions f_i are constructed from the data. We will therefore set $n = m$ and use one basis function per data point. The objective is to minimize the least-squares cost

$$J = \sum_{i=1}^{m} (x_1 f_1(s^i) + \cdots + x_n f_n(s^i) - g^i)^2$$

Let $g \in \mathbb{R}^m$ be given by $g = (g^1, g^2, \ldots, g^m)$.

a) You are given a function $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$. We will use it to make m basis functions

$$f_j(t) = k(t, s^j) \qquad \text{for all } t \in \mathbb{R}^d$$

where s^j is the independent variable of the j'th data point (not s to the power of j). We have one basis function for each data point. Define the matrix $Q \in \mathbb{R}^{m\times m}$

$$Q_{ij} = k(s^i, s^j) \qquad \text{for all } i, j = 1, \ldots, m$$

Formulate the problem of finding $x \in \mathbb{R}^m$ to minimize the least-squares cost in terms of Q and g.

b) Under what conditions on Q will there exist a unique solution x to the above optimization problem?

c) We are given $q \in \mathbb{R}^d$ and we would like to find the value of the predictor $\hat{g}(q)$. Define the vector $z \in \mathbb{R}^m$ by

$$z_i = k(q, s^i) \qquad \text{for all} i = 1, \dots, m$$

Find an expression for the predictor $\hat{g}(q)$ in terms of g, Q, and z.

d) Try this using the following function

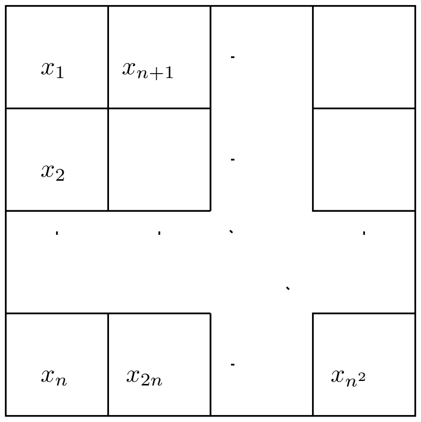
$$k(s,t) = \exp(-\gamma \|s - t\|^2)$$

with data

$$s = (0.2, 0.3, 0.4, 0.5, 0.6, 0.8)$$
$$g = (-1, -1.5, 0.4, 1, 1.2, 1)$$

Here $d = 1$, $n = 6$ and $m = 6$. Try values $\gamma = 10, 100, 1000$. Plot the data points and the fitted function $\hat{g}$ versus s on the interval $[0,1]$ in each case. Sample the interval $[0,1]$ appropriately so that your plot shows that value of $\hat{g}$ not just at the sample points but also in between the sample points.

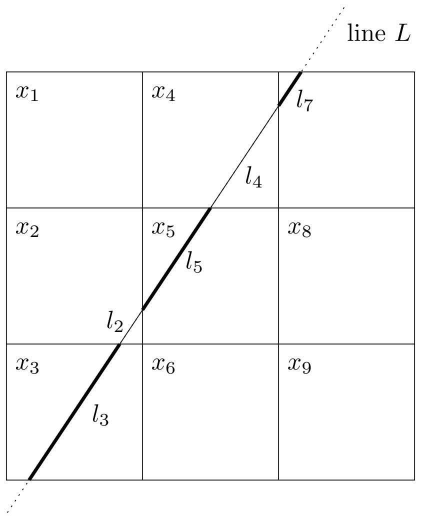
6.741. Image reconstruction from line integrals. In this problem we explore a simple version of a tomography problem. We consider a square region, which we divide into an $n \times n$ array of square pixels, as shown below.



The pixels are indexed column first, by a single index i ranging from 1 to n^2, as shown above. We are interested in some physical property such as density (say) which varies over the region. To simplify things, we'll assume that the density is constant inside each pixel, and we denote by x_i the density in pixel i, $i = 1, \dots, n^2$. Thus, $x \in \mathbb{R}^{n^2}$ is a vector that describes the density across the rectangular array of pixels. The problem is to estimate the vector of densities x, from a set of sensor measurements that we now describe. Each sensor measurement is a *line* *integral* of the density over a line L. In addition, each measurement is corrupted by a (small) noise term. In other words, the sensor measurement for line L is given by

$$\sum_{i=1}^{n^2} l_i x_i + v,$$

where l_i is the length of the intersection of line L with pixel i (or zero if they don't intersect), and v is a (small) measurement noise. This is illustrated below for a problem with $n = 3$. In this example, we have $l_1 = l_6 = l_8 = l_9 = 0$.



Now suppose we have N line integral measurements, associated with lines $L_1, \ldots, L_N$. From these measurements, we want to estimate the vector of densities x. The lines are characterized by the intersection lengths

$$l_{ij}, \quad i = 1, \ldots, n^2, \quad j = 1, \ldots, N,$$

where l_{ij} gives the length of the intersection of line L_j with pixel i. Then, the whole set of measurements forms a vector $y \in \mathbb{R}^N$ whose elements are given by

$$y_j = \sum_{i=1}^{n^2} l_{ij} x_i + v_j, \quad j = 1, \ldots, N.$$

And now the problem: you will reconstruct the pixel densities x from the line integral measurements y. The class webpage contains the file `tomo_data.json`, which contains the following variables:

- `N`, the number of measurements (N),
- `npixels`, the side length in pixels of the square region (n),
- `y`, a vector with the line integrals y_j, $j = 1, \ldots, N$,

- `line_pixel_lengths`, an $n^2 \times N$ matrix containing the intersection lengths l_{ij} of each pixel $i = 1, \ldots, n^2$ (ordered column-first as in the above diagram) and each line $j = 1, \ldots, N$,
- `lines_d`, a vector containing the displacement (distance from the center of the region in pixel lengths) d_j of each line $j = 1, \ldots, N$, and
- `lines_theta`, a vector containing the angles θ_j of each line $j = 1, \ldots, N$.

(You shouldn't need `lines_d` or `lines_theta`, but we're providing them to give you some idea of how the data was generated. Similarly, the file `tmeasure.jl` shows how we computed the measurements, but you don't need it or anything in it to solve the problem. The variable `line_pixel_lengths` was computed using the function in this file.)

Use this information to find x, and display it as an image (of n by n pixels). You'll know you have it right.

Julia hints:

- The `reshape` function might help with converting between vectors and matrices, for example, `A = reshape(v, m, n)` will convert a vector with $v = mn$ elements into an $m \times n$ matrix.
- To display a matrix `A` as a grayscale image, you can use: (or any method that works for you)
  ```
  heatmap(A, yflip=true, aspect_ratio=:equal, color=:gist_gray,
          cbar=:none, framestyle=:none)
  ```
  You'll need to have loaded the JuliaPlots package with `using Plots` to access the `heatmap` function. (The `yflip` argument gets it to plot the origin in the top-left rather than the bottom-left.)

Note: While irrelevant to your solution, this is actually a simple version of *tomography*, best known for its application in medical imaging as the CAT scan. If an *x-ray* gets attenuated at rate x_i in pixel i (a little piece of a cross-section of your body), the j-th measurement is

$$z_j = \prod_{i=1}^{n^2} e^{-x_i l_{ij}},$$

with the l_{ij} as before. Now define $y_j = -\log z_j$, and we get

$$y_j = \sum_{i=1}^{n^2} x_i l_{ij}.$$

6.790. Identifying a system from input/output data. We consider the standard setup:

$$y = Ax + v,$$

where $A \in \mathbb{R}^{m\times n}$, $x \in \mathbb{R}^n$ is the input vector, $y \in \mathbb{R}^m$ is the output vector, and $v \in \mathbb{R}^m$ is the noise or disturbance. We consider here the problem of estimating the matrix A, given some input/output data. Specifically, we are given the following:

$$x^{(1)}, \dots, x^{(N)} \in \mathbb{R}^n, \qquad y^{(1)}, \dots, y^{(N)} \in \mathbb{R}^m.$$

These represent N samples or observations of the input and output, respectively, possibly corrupted by noise. In other words, we have

$$y^{(k)} = Ax^{(k)} + v^{(k)}, \quad k = 1, \dots, N,$$

where $v^{(k)}$ are assumed to be small. The problem is to estimate the (coefficients of the) matrix A, based on the given input/output data. You will use a least-squares criterion to form an estimate $\hat{A}$ of A. Specifically, you will choose as your estimate $\hat{A}$ the matrix that minimizes the quantity

$$J = \sum_{k=1}^{N} \|Ax^{(k)} - y^{(k)}\|^2$$

over A.

a) Explain how to do this. If you need to make an assumption about the input/output data to make your method work, state it clearly. You may want to use the matrices $X \in \mathbb{R}^{n\times N}$ and $Y \in \mathbb{R}^{m\times N}$ given by

$$X = \begin{bmatrix} x^{(1)} & \cdots & x^{(N)} \end{bmatrix}, \qquad Y = \begin{bmatrix} y^{(1)} & \cdots & y^{(N)} \end{bmatrix}$$

in your solution.

b) On the course web site you will find some input/output data for an instance of this problem in the file `sysid_data.json`. Executing this Julia file will assign values to m, n, and N, and create two matrices that contain the input and output data, respectively. The $n \times N$ matrix variable `X` contains the input data $x^{(1)}, \dots, x^{(N)}$ (*i.e.*, the first column of `X` contains $x^{(1)}$, etc.). Similarly, the $m \times N$ matrix `Y` contains the output data $y^{(1)}, \dots, y^{(N)}$. You must give your final estimate $\hat{A}$, your source code, and also give an explanation of what you did.

6.1240. Iteratively reweighted least squares for 1-norm approximation. In an ordinary least squares problem, we are given $A \in \mathbb{R}^{m\times n}$ (skinny and full rank) and $y \in \mathbb{R}^m$, and we choose $x \in \mathbb{R}^n$ in order to minimize

$$\|Ax - y\|_2^2 = \sum_{i=1}^{m} (\tilde{a}_i^T x - y_i)^2.$$

Note that the penalty that we assign to a measurement error does not depend on the sensor from which the measurement was taken. However, this is not always the right thing to do: if we believe that one sensor is more accurate than another, we might want to assign a larger penalty to an error in the measurement from the more accurate sensor. We can account for differences in the accuracies of our sensors by assigning sensor i a weight $w_i > 0$, and then minimizing

$$\sum_{i=1}^{m} w_i(\tilde{a}_i^T x - y_i)^2.$$

By giving larger weights to more accurate sensors, we can account for differences in the precision of our sensors.

a) *Weighted least squares.* Explain how to choose x in order to minimize

$$\sum_{i=1}^{m} w_i(\tilde{a}_i^T x - y_i)^2,$$

where the weights $w_1, \dots, w_m > 0$ are given.

b) *Iteratively reweighted least squares for ℓ_1-norm approximation.* Consider a cost function of the form

$$\sum_{i=1}^{m} w_i(x)(\tilde{a}_i^T x - y_i)^2. \tag{1}$$

One heuristic for minimizing a cost function of the form given in (1) is *iteratively reweighted least squares*, which works as follows. First, we choose an initial point $x^{(0)} \in \mathbb{R}^n$. Then, we generate a sequence of points $x^{(1)}, x^{(2)}, \dots \in \mathbb{R}^n$ by choosing $x^{(k+1)}$ in order to minimize

$$\sum_{i=1}^{m} w_i(x^{(k)})(\tilde{a}_i^T x^{(k+1)} - y_i)^2.$$

Each step of this algorithm involves updating our weights, and solving a weighted least squares problem. Suppose we want to use this method to solve minimize the ℓ_1-norm approximation error, which is defined to be

$$\|Ax - y\|_1 = \sum_{i=1}^{m} |\tilde{a}_i^T x - y_i|,$$

where the matrix $A \in \mathbb{R}^{m \times n}$ and the vector $y \in \mathbb{R}^m$ are given. How should we choose the weights $w_i(x)$ to make the cost function in (1) equal to the ℓ_1-norm approximation error?

c) *Numerical example.* The file `l1_irwls_data.json` contains data $(t_1, y_1), \dots, (t_m, y_m)$. We want to fit an affine model to this data:

$$y_i = x_1 + x_2 t_i, \qquad i = 1, \dots, m.$$

Choose $x^{(0)}$ to be the vector of least-squares parameter estimates: that is, choose $x^{(0)}$ in order to minimize

$$\sum_{i=1}^{m} ((x_1^{(0)} + x_2^{(0)} t_i) - y_i)^2.$$

Generate $x^{(1)}, x^{(2)}, \ldots$ using iteratively reweighted least squares for ℓ_1-norm approximation. You can stop generating iterates when $\|x^{(k+1)} - x^{(k)}\| < 10^{-6}$. Report your values of $x^{(0)}$ and the final $x^{(k)}$ in your sequence of points. Draw a scatterplot of the data points (t_i, y_i). Add the fitted lines corresponding to $x^{(0)}$ and the final $x^{(k)}$ to your scatterplot. What do you observe?

Remark. Suppose we fit the least-squares line to some data. Then, a point that is very far from the least-squares line may be an *outlier*: that is, a point that does not seem to follow the same model as the rest of the data. Because such points may not follow the same model as the rest of data, it may make sense to give such points less weight. This idea is the intuition behind iteratively reweighted least squares for ℓ_1-norm approximation.