02-512/02-712 Computational Methods for Biological Modeling and Simulation
Lecturers: Russell Schwartz and Oana Carja
Tuesdays and Thursdays 2:00pm-3:20pm HH B131

Problem Set 2

Assigned 9/21/23, Due 10/5/23

Problem 1 (15 points): The goal of this problem will be to attempt to formulate a few variations on a regression problem in terms of linear systems. For each variant, formally describe a linear system for the biological problem. Then, identify whether the system is overdetermined, underdetermined, or full rank. Finally, suggest an algorithm you could use to solve for that variant.

We will assume that our goal is to optimize growth conditions for a bacterium we are trying to culture. We assume that we have a set of environmental conditions we can control, such as temperature at which they are grown, pressure, amounts of various chemicals in the growth media, etc. We can represent these conditions as a set of real-valued variables $c_1, \ldots, c_n$. We would like to learn a model of how growth rate G depends on these parameters.

a. Suppose we want to create a linear regression model, assuming a baseline growth rate g_0 and assuming that growth rate depends linearly on each of the n parameters we can control. We would like to learn this model of growth rate by running $n+1$ experiments with randomly chosen parameter settings. (You can assume the combinations of experiments are all linearly independent.)

b. Suppose we now use the same setup as in part a, but instead do $2n$ experiments. (You can again assume the combinations of experiments are all linearly independent.)

c. Let us suppose we keep $2n$ experiments but allow the model to fit all possible quadratic terms ($e.g., c_1^2, c_1c_2$) in addition to linear terms. (You can again assume the combinations of experiments are all linearly independent.)

d. Now, let us keep the quadratic model but increase the number of experiments to n^2. (You can again assume the combinations of experiments are all linearly independent. We will also assume here that $n \geq 6$.)

Problem 2 (20 points): In this problem, we will examine how various zero-finding methods perform on a single system. We will examine this in the context of a toy 1-dimensional version of a molecular docking problem. We will assume that we are trying to dock a small molecule into a binding pocket of a protein. We will suppose that the forces acting on the molecule can be approximately divided into a steric repulsion force preventing the molecule from moving too deeply into the molecule, a charge force drawing it in, and a generic effect of entropy biasing it to move out of the pocket. We model these by the following function:

$$\frac{4}{x^7} - \frac{3}{x^2} + 0.1$$

where x is the distance from the bottom of the binding pocket. We want to find the optimal position for the molecule to sit, i.e., where the forces balance to a net zero force on the molecule. You can assume we have established this is somewhere on the range $x \in [1, 3]$.

a. Perform three steps of bisection search to solve for x from the starting interval [1,3], showing your work (i.e., the translation x and the net force at the endpoints and midpoint of each step). Each time you derive a new midpoint counts as one step. Provide your final estimate (the midpoint of the last interval) and the forward and maximum possible backward errors at that point.

b. Again assuming the forces balance somewhere on the range $x \in [1, 3]$, perform three steps of secant method search, showing your work as in part a. Each time you derive a new midpoint counts as one step. Provide your final estimate (the midpoint of the last interval) and the forward and maximum possible backward errors at that point.

c. Perform three steps of Newton-Raphson search to solve for the point at which the forces balance, showing your work. We will assume an initial guess of $x = 1$. Each time you derive a new estimated x counts as one iteration. Provide your final estimate, its forward error, and an estimate of the backward error at that point.

Problem 3 (20 points): In this problem, we will use linear programming for a hypothetical application in experimental study design. Let us suppose we want to conduct a sequencing study of changes in gene expression across tissues. We have different sequencing methods available to us — let us say we can do single-cell RNA-seq (a), bulk RNA-seq (b), or spatial RNA-seq (c). Each we will assume has some intrinsic utility for solving our problem that we can approximately quantify. But we will also assume that we have constraints on the designs we can conduct. We have a finite amount of money to spend on sequencing and each technology has a different cost per unit of sequence. We have a finite amount of time we can spend to get the sequencing done and each has a different amount of time we need per unit of sequence. And we have a finite amount of tissue to analyze and each technology has different needs with respect to how much tissue we need per unit of sequence to apply it. Let us suppose the following table encodes these factors:

technology	cost ($k)	time (days)	tissue (g)	utility
a: single-cell	2	1	3	1
b: bulk	1	1	1	2
c: spatial	3	2	1	5

Our goal will be to find the optimal mixture of technologies to maximize the utility of our data set within our resource limits. Assume we have 10 thousand dollars, 7 days, and 12 grams of tissue to work with.

a. We can pose this as a linear program to solve for variables a, b, and c representing how much we will order of each sequencing technology so as to maximize utility without exceeding any of our resource limits. Provide the constraints and objective function to define this problem.

b. Convert the program into standard form.

c. Use the simplex method to find the optimal solution, showing your work. You can use no sequencing ($a = 0, b = 0, c = 0$) as an initial starting point.

d. What is the optimal assignment of sequencing technologies you derived and the utility of that choice?

e. How much of each of the three resources (money, time, and tissue) would you use for your optimal result?

Problem 4 (25 points): This is a programming problem in which we will use some of what we have learned about numerical optimization for an application in bioimage analysis. We will look to solve the problem of recognizing an object in an image. This is something we might want to solve to find instances of a particular kind of organelle in cellular images, or to find a particular molecular complex in cryo-electron microscopy images. We will assume here that we are provided an image and a template and we want to find instances of things close to the template in the image. To keep the problem relatively simple, we will only consider fitting translationally, i.e., finding the right x and y coordinates of the template in the image, and will not consider that the template might need to be rotated or scaled to fit it to the image.

Let us assume that our image is is represented as an $M \times N$ matrix A, where element A_{ij} is the intensity of the pixel centered on position (i, j), and the template is similarly represented as an $m \times n$ matrix T for $m \leq M, n \leq N$, where T_{ij} is the intensity of the object at position (i, j) of the template. Our goal will be to fit the template to the image by finding translations (x, y) of the template relative to the image to align them. You can assume that both images define the upper left corner to be (0,0) and that (x, y) are therefore positive values between $(0, 0)$ and $(M - m, N - n)$. Note that we are assuming the translations might be real-valued, i.e., that the object might not line up with pixel boundaries. We will assume that we can only recognize objects that are found entirely within the image and define an objective function based on sum-of-squares difference between the template and the image.

We can define the intensity of the image at a real-valued point (x', y') by the interpolation formula:

$$I(A, x', y') = \frac{\sum_{p=1}^{M} \sum_{q=1}^{N} a_{pq} e^{-\sigma^2((x'-p)^2+(y'-q)^2)}}{\sum_{p=1}^{M} \sum_{q=1}^{N} e^{-\sigma^2((x'-p)^2+(y'-q)^2)}}$$

where σ^2 is a scaling constant we will arbitrarily set to 5 for this problem.

Then we can pose the optimization problem for aligning the template to the image using the following objective function:

$$f(A, T, x, y) = \sum_{i=1}^{m} \sum_{j=1}^{n} (t_{ij} - I(A, i + x, j + y))^2$$

a. We may decide this objective is too messy to deal with analytical derivatives, so provide formulas for the terms of the gradient using second order finite difference approximations with step sizes dx and dy.

b. Provide second order numerical approximations to the terms of the Hessian given step sizes dx and dy.

c. We can turn the N-R optimizer into a global optimizer by locally optimizing over a grid of values of x and y, essentially trying every possible starting (x_0,y_0) with some increments δx and δy and taking the best one found. For sufficiently small increments, we can prove that at least one point must be within the radius of convergence of the global optimum. Provide pseudocode for a pseudo-global optimizer, using Newton-Raphson for local optimization, for this objective function. You can assume it is enough to run it for ten steps and it will either converge or blow up by selecting an (x, y) outside the allowed range. Hint: For a 2x2 system of equations, the solution to

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

is

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} (a_{22}b_1 - a_{12}b_2)/(a_{11}a_{22} - a_{12}a_{21}) \\ (a_{11}b_2 - a_{21}b_1)/(a_{11}a_{22} - a_{12}a_{21}) \end{bmatrix}.$$

d. Write code implementing your N-R optimizer and this peudoglobal wrapper for it to fit an object to a template and return the best-fitting point. Your code should read an input file specified as follows:

$\delta x\ \delta y\ dx\ dy\ \sigma^2$
$M\ N$
$a_{11}\ a_{12}\ \ldots\ a_{1N}$
$a_{21}\ a_{12}\ \ldots\ a_{1N}$
$\vdots$
$a_{M1}\ a_{M2}\ \ldots\ a_{MN}$
$m\ n$
$t_{11}\ \ldots\ t_{1N}$
$t_{21}\ \ldots\ t_{1N}$
$\vdots$
$t_{M1}\ \ldots\ t_{MN}$

Your code should return the best-fitting (x, y) it can find, or (-1,-1) if it cannot find any point that does not diverge. (If you would prefer to work with analytical rather than numerical derivatives in your code, that is fine as long as they are correct.)

e. Test your code on the provided sample inputs and return the answers.

***Problem 5 (20 points):** In this problem, we will consider some alternative solutions to the problem of solving for overdetermined linear systems. In class, we saw that the most common way to handle this is to find a least-squares best fit to the overdetermined linear system $A\vec{x} = \vec{b}$ by solving instead for the full rank linear system $(A^TA)\vec{x} = A^T\vec{b}$. This is not the only way we might pose the problem of finding the best solution when there is no $\vec{x}$ that satisfies every constraint perfectly, however.

a. Let us suppose we want to change the objective function to optimize for the L_1 norm of the residual vector rather than the L_2 norm. That is, we want to minimize the sum of absolute values of the elements of the residual vector $A\vec{x} - \vec{b}$, $|A\vec{x} - \vec{b}|$, rather than the sum of squares $||A\vec{x} - \vec{b}||$. Pose this as a linear program, providing any constraints and objective function to cast it in this form, and suggest an algorithm to solve it. (Hint: You will need some auxiliary variables and appropriate constraints to represent absolute value terms.)

b. Another thing we might try is a regularized version of the least squares regression problem. That is, we approximately optimize for the least-squares problem but further penalize solutions that are "bad" by some measure. A typical way of doing that is to add a penalty based on the sum of absolute values of elements in x we are trying to learn, $\sum_i |x_i|$. That is, our objective function will be the least squares objective plus a term of the form $\lambda \sum_i |x_i|$, where λ is a regularization constant. This will tend to favor solutions where many elements of $\vec{x}$ are zero, which is a desirable property if we want to favor simpler models or remove variables that might just be fitting to noise in our data. Pose this as a quadratic programming problem, providing the constraints and objective function, and suggest an algorithm to solve for it.

c. Show that the objective function in part b is convex, and thus that we can optimally solve for the problem as posed. (Hint: Think about how the convexity of this problem is related to the convexity of the standard least squares optimization without L_1 regularization.)

*Recall that the starred problems are only for the 02-712 students.