18.S096 PSET 2, IAP 2023

Problem 1 (5+5+5 points)

Suppose that $A(p)$ takes a vector $p \in \mathbb{R}^{n-1}$ and returns the $n \times n$ tridiagonal real-symmetric matrix

$$A(p) = \begin{pmatrix} a_1 & p_1 & & & \\ p_1 & a_2 & p_2 & & \\ & p_2 & \ddots & \ddots & \\ & & \ddots & a_{n-1} & p_{n-1} \\ & & & p_{n-1} & a_n \end{pmatrix},$$

where $a \in \mathbb{R}^{n-1}$ is some constant vector. Now, define a scalar-valued function $f(p)$ by

$$f(p) = \left(c^T A(p)^{-1} b\right)^2$$

for some constant vectors $b, c \in \mathbb{R}^n$ (assuming we choose p and a so that A is invertible). Note that, in practice $A(p)^{-1}b$ is *not* computed by explicitly inverting the matrix A—instead, it can be computed in $\Theta(n)$ (i.e., roughly proportional to n) arithmetic operations using Gaussian elimination that takes advantage of the "sparsity" of A (the pattern of zero entries), a "tridiagonal solve".

(a) Write down a formula for computing $\partial f / \partial p_1$ (in terms of matrix–vector products and matrix inverses). (Hint: once you know df in terms of dA, you can get $\partial f / \partial p_1$ by "dividing" both sides by ∂p_1, so that dA becomes $\partial A / \partial p_1$.)

(b) Outline a sequence of steps to compute both f and ∇f (with respect to p) using only *two* tridiagonal solves $x = A^{-1}b$ and an "adjoint" solve $v = A^{-1}$(something), plus $\Theta(n)$ (i.e., roughly proportional to n) additional arithmetic operations.

(c) Write a program implementing your ∇f procedure (in Julia, Python, Matlab, or any language you want) from the previous part. (You don't need to use a fancy tridiagonal solve if you don't know how to do this in your language; you can solve A^{-1}(vector) inefficiently if needed using your favorite matrix libraries.) Implement a finite-difference test: Choose a, b, c, p at random, and check that $\nabla f \cdot \delta p \approx f(p + \delta p) - f(p)$ (to a few digits) for a randomly chosen small δp.

Problem 2 (5+5 points)

Suppose that we have a two-argument function $f(x, y)$, where x, y and f may belong to arbitrary vector (Banach) spaces. Let's define "partial" derivatives f_x and f_y (also denoted $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial y}$) by the linearization:

$$df = f(x + dx, y + dy) - f(x, y) = f_x(x, y)[dx] + f_y(x, y)[dy],$$

implicitly dropping higher-order terms as usual. Compute the partial derivatives of the following functions:

(a) $f(A, x) = A^{-1}x$ for $n \times n$ matrices $A \in \mathbb{R}^{n\times n}$ and vectors $x \in \mathbb{R}^n$: give f_A as a linear operator, and f_x as a Jacobian matrix.

(b) $f(A, B) = \mathrm{tr}(A^T BA)$, for matrices $A, B \in \mathbb{R}^{n\times n}$: give the gradients $\nabla_A f$ and $\nabla_B f$ such that $f_A[dA] = \nabla_A f \cdot dA$ and $f_B[dB] = \nabla_B f \cdot dB$ under the Frobenius inner product $X \cdot Y = \mathrm{tr}(X^T Y) = \mathrm{tr}(Y^T X)$.

Problem 3 (5+5 points)

If S is an $m\times m$ real-symmetric matrix with a "simple" (multiplicity = 1) eigenvalue λ and corresponding eigenvector q ($Sq = \lambda q$), normalized to $q^T q = 1$, then the "Hellman–Feynman theorem" states that $d\lambda = q^T dS\, q$ for a change dS in the matrix S.

(a) Derive the Hellman–Feynman theorem by considering the differentials of both sides of the equations $d(\lambda = q^T Sq)$ and $d(q^T q = 1)$.

(b) What is the gradient $\nabla\lambda$ with respect to S, for the usual Frobenius inner product $\nabla\lambda \cdot dS = \mathrm{tr}((\nabla\lambda)^T dS)$

Problem 4 (6+6 points)

The Jacobian determinant (sometimes called simply "the Jacobian," clashing with the concept of the Jacobian matrix) is the determinant of the Jacobian matrix. Specifically if $f(x)$ is a function from $\mathbb{R}^n$ to $\mathbb{R}^n$ and $(\frac{\partial f_i}{\partial x_j})_{1\le i,j\le n}$ is the Jacobian matrix $f'(x)$, then its determinant $\det f'(x)$ is the Jacobian determinant. Sometimes we take the absoute value and not worry too much about the sign.

(a) The Jacobian determinant represents the local scaling of volume. Compute the Jacobian determinant of the hyperbolic rotation defined in Pset 1, problem 1b, in simplest form. Use this to describe how a little square around a point generally transforms with a hyperbolic rotation.

(b) There are many ways to equivalently take a scalar function $f(\alpha)$ and extend it to a matrix function $F(M)$, which takes in a square matrix and returns a square matrix of the same size.

The simplest is to define $f(M) = Xf(\Lambda)X^{-1}$, where $M = X\Lambda X^{-1}$ is an eigen-decomposition of M (and use continuity to include non-diagonalizable matrices). Here, $f(\Lambda)$ denotes the application of a scalar function $f(\lambda)$ to the eigenvalues λ (on the diagonal of Λ). (e.g., you've probably seen e^M defined in terms of e^λ.)

One could then write $f'(M)$ as an explicit $n^2\times n^2$ Jacobian matrix (e.g. via $\mathrm{vec}(dM)$ and Kronecker products), and could then compute its determinant.

(i) Write a computer program (in any language) to find the 9×9 Jacobian matrix of $f(M)$ and then the Jacobian determinant by either finite differences or by using automatic differentiation, for $f(\lambda)$ being e^λ, λ^2, and $\sin(\lambda)$ on the 3×3 matrix $M = [0\ 1\ 4; 1\ 0\ 1; 4\ 1\ 0]$ with entries $M_{i,j} = (i - j)^2$.

(ii) Compare with the following known theoretical formula for the Jacobian determinant for a scalar function $f(\lambda)$ applied to a diagonalizable matrix M, in terms of M's eigenvalues λ:

$$\frac{\prod_{i<j} |f(\lambda_i) - f(\lambda_j)|^2}{\prod_{i<j} |\lambda_i - \lambda_j|^2} \prod_i f'(\lambda_i)$$

18.S096 Matrix Calculus for Machine Learning and Beyond
Independent Activities Period (IAP) 2023