Homework 2 Solutions

January 27, 2022

Problem 1: 10 points

Let $h(x) = g(f(x))$ Then, by the chain rule, $h'(x) = g'(f(x))f'(x)$ (this the $n \times n$ Jacobian matrix). But since $g = f^{-1}$, we also have $h(x) = x = I_n x$ and hence $h'(x) = I_n$, the $n \times n$ identity matrix. So, $g'(f(x))f'(x) = I_n$, and thus $\boxed{g'(f(x)) = f'(x)^{-1}}$, or equivalently $g'(y) = f'(g(y))^{-1}$ (either way is fine): the Jacobians of f and g are **inverses**.

(Be careful to evaluate the functions with the correct arguments! $f'(x) \neq g'(x)^{-1}$!)

This is closely related to the "inverse function theorem": a differentiable function $f(x)$ is locally invertible (near x) if and only if its Jacobian is invertible, i.e. when $\det f'(x) \neq 0$.

Problem 2: 10+5 points

Consider $f(A) = \sqrt{A}$ where A is an $n \times n$ matrix.

a) Following the hint, we realize that $f(A)$ is the inverse of $g(B) = B^2$, so we just need to compute the Jacobian of $g(f(B))$ and invert it. From class, $dg = B\,dB + dB\,B \implies \text{vec}(dg) = (I_n \otimes B + B^T \otimes I_n)\,\text{vec}(dB)$, and hence (setting $B = f(A) = \sqrt{A}$ as in problem 1)

$$\text{vec}(df) = \text{vec}(f'(A)dA) = \boxed{(I_n \otimes \sqrt{A} + \sqrt{A}^T \otimes I_n)^{-1}}\,\text{vec}(dA)\,,$$

where the boxed term is the Jacobian.

b) We'll check our formula numerically against a finite-difference approximation, with a random positive-definite A: **See Julia notebook**.

As an even easier check, it's also nice (but not required) to try the 1×1 case, which should agree with the scalar derivative $\sqrt{a}' = 1/2\sqrt{a}$. For 1×1 matrices, $\otimes$ is just ordinary multiplication, so we get $(1 \otimes \sqrt{a} + \sqrt{a}^T \otimes 1)^{-1} = 1/2\sqrt{a}$ as expected.

Problem 3: $(3 \times 5) + 5 + 5$ points

Here, we are computing

$$f(p) = g(\underbrace{B(\underbrace{A(p)^{-1}a}_{x})^{-1}b}_{y})\,,$$

where $A(p) = A_0 + \text{diagm}(p)$, $B(x) = B_0 + \text{diagm}(x \mathbin{.*} x)$, and $g(y) = y^T F y$.

Now, suppose that we want to optimize $f(p)$ as a function of the parameters p that went into the first step. We need to compute the gradient ∇f for any kind of large-scale problem. If n is huge, though, we must be careful to do this efficiently, using "reverse-mode" or "adjoint" calculations in which we apply the chain rule from left to right.

a) Let us start applying the chain rule to df, with the rule from class for the derivative of a matrix inverse:

$$\begin{aligned} df &= g'(y)\,\underbrace{d(B^{-1})b}_{dy} \\ &= -g'(y)B^{-1}dB\,\underbrace{B^{-1}b}_{y} \\ &= \underbrace{\underbrace{-\underbrace{2y^T F}_{g'(y)}\,B^{-1}}_{u^T}\,dB}_{v^T}\,y\,, \end{aligned}$$

where the labels indicate our left-to-right order of evaluation:

i) $\boxed{g'(y) = 2y^T F}$

ii) Let $u^T = -g'(y)B^{-1}$, i.e. we solve the first **adjoint equation** $\boxed{B^T u = -g'(y)^T}$.

iii) Linearizing $B(x)$ yields[1] $dB = 2\,\mathrm{diagm}(x \mathbin{.*} dx)$. Hence $v^T = u^T dB$ yields the diagonal-matrix/elementwise product $v = u \mathbin{.*} 2x \mathbin{.*} dx$, giving $df = v^T y$.

To get to the next step, the trick is to write $df = v^T y$ as some w^T multiplying dx from the *left*.[2] But this is easy if we write it out componentwise: $v_i = 2u_i x_i dx_i$, and $df = \sum_i v_i y_i = \sum_i 2u_i x_i y_i dx_i$. Hence $df = w^T dx$ where $\boxed{w = 2u \mathbin{.*} x \mathbin{.*} y}$.

iv) $df = w^T dx = w^T d(A^{-1})a = -w^T A^{-1}\,dA\,A^{-1}a = -w^T A^{-1}\,dA\,x$. So, multiplying left-to-right, we let $z^T = -w^T A^{-1}$, i.e. we solve a second **adjoint equation** $\boxed{A^T z = -w}$ for z.

v) Finally, $df = z^T dA\,x$. Again linearizing, $dA = \mathrm{diagm}(dp)$, so $df = z^T(dp \mathbin{.*} x) = \sum_k z_k dp_k x_k$, or equivalently $\frac{df}{dp_k} = z_k x_k$, or equivalently $\boxed{\nabla f = z \mathbin{.*} x}$.

Each of the boxed steps above consists of either a simple matrix–vector or vector–vector product, or in steps (ii) and (iv) we perform a *single* $n \times n$ linear "adjoint" solve with B^T and A^T, respectively, similar to f itself.[3]

b) We'll check answer numerically against finite differences (for some randomly chosen A_0, B_0, a, b, F): **See Julia notebook**.

c) We'll also check our answer against the result of a reverse-mode AD software (Zygote in Julia): **See Julia notebook**.

Problem 4: 10×3 points

a) Write down some 4×4 matrix that is not the Kronecker product of two 2×2 matrices. Convince us this is true.

Answer: Anything where one 2×2 corner block is not a multiple of another 2×2 corner block would work. For example:

$$\boxed{\begin{pmatrix} 1 & 2 & 0 & 0 \\ 3 & 4 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{pmatrix}}\,.$$

The reason for this is that a 2×2 Kronecker product is of the form:

$$\begin{pmatrix} a & c \\ b & d \end{pmatrix} \otimes B = \begin{pmatrix} a\,B & c\,B \\ b\,B & d\,B \end{pmatrix}\,,$$

so every 2×2 corner block of the result *must* be a scalar multiple of the same matrix B.

[1] Since diagm is linear, $d(\mathrm{diagm}(\cdots)) = \mathrm{diagm}(d(\cdots))$, and we saw how to differentiate elementwise functions in pset 1.

[2] It is not an accident that we can do this! df is a scalar linear function of dx, and *all* such functions can be written as dot products $w^T dx$.

[3] In fact, we can probably re-use computations between the forward and adjoint solves. Most obviously, if we compute A^{-1} explicitly, then $(A^T)^{-1} = (A^{-1})^T$. In practice, one rarely computes matrix inverses, but if one has an LU factorization $A = LU$, then $A^T = U^T L^T$ and we can re-use the same triangular factors to solve $A^T z = -w$. This is especially important if A and B are large and sparse matrices, because in this case the LU factors are often sparse but the inverses are not.

b) Prove that if A $(m \times m)$ and B $(n \times n)$ are orthogonal (i.e., $A^T A = I_m$ and $B^T B = I_n$) then $A \otimes B$ $(mn \times mn)$ is orthogonal.

Answer: Let us check whether $A \otimes B$ satisfies

$$(A \otimes B)^T (A \otimes B) = I_{mn}\,.$$

Using Kronecker-product properties, $(A \otimes B)^T = A^T \otimes B^T$ (no reversal through the Kronecker product). Also, matrix multiplication of Kronecker products equals Kronecker products of matrix multiplies:

$$(A^T \otimes B^T) \otimes (A \otimes B) = (A^T A \otimes B^T B) = I_m \otimes I_n = I_{mn}.$$

c) If $f(A) = e^A = \sum_{k=0}^{\infty} A^k/k!$, write down a power series involving Kronecker products for the Jacobian $f'(A)$. Check your answer with a numerical example, e.g. against finite differences.

Answer:

$$df = \sum_{k=0}^{\infty} \frac{d(A^k)}{k!} = \sum_{k=0}^{\infty} \frac{1}{k!} \sum_{\ell=0}^{k-1} A^\ell dA A^{k-1-\ell}.$$

We can write this as

$$\mathrm{vec}(df) = \underbrace{\boxed{\sum_{k=0}^{\infty} \frac{1}{k!} \left(\sum_{\ell=0}^{k-1} (A^T)^{k-1-\ell} \otimes A^\ell \right)}}_{\text{Jacobian}} \mathrm{vec}(dA)\,.$$

See the accompanying Julia notebook for the numerical validation.