Homework 1 Solutions

January 27, 2024

See also the accompanying Julia notebook for computational solutions.

Problem 1 (5 points)

Start reading the draft course notes (linked from https://github.com/mitmath/matrixcalc/). Find a place that you found confusing, and write a paragraph explaining the source of your confusion and (ideally) suggesting a possible improvement.

(Any other corrections/comments are welcome, too.)

Solution:

Student-dependent, but full marks if clearly written and explained.

Problem 2 (5 points)

A directional derivative of $f(x)$ in a direction v is sometimes described as the derivative $\frac{d}{d\alpha} f(x+\alpha v)\big|_{\alpha=0}$, where $\alpha \in \mathbb{R}$ is a scalar; that is, it is $g'(0)$ for $g(\alpha) = f(x+\alpha v)$. If $f(x)$ is a function from some input vector space $x \in X$ to some output vector space $f(x) \in Y$ with a derivative $f'(x)$ as defined in class, apply the chain rule to obtain this $g'(0)$ (for some $v \in X$) in terms of f'.

Solution:

Let $h(\alpha) = x + \alpha v$. We have $g(\alpha) = f(h(\alpha))$, so by the chain rule we must have $g'(0)d\alpha = f'(h(0))[h'(0)d\alpha] = f'(x)[h'(0)]d\alpha$ (the scalar $d\alpha$ can be pulled out by linearity). But $dh = h'(\alpha)[d\alpha] = d\alpha\, v$, so h' is simply the linear operator that multiplies $d\alpha$ by v, i.e. $h' = v$. So, we have:

$$g'(0) = f'(x)[h'(0)] = \boxed{f'(x)[v]},$$

which is the same as the "directional derivative" defined in class.

For example, if x is a column vector and $f(x)$ is a scalar, so that $f' = (\nabla f)^T$, this reduces to $(\nabla f)^T v$ (the dot product of v with the gradient), which is another formula you may have seen.

Problem 3 (3+3+3+3+3 points)

Find the derivatives f' of the following functions. If f maps column vectors to scalars, give ∇f (so that $f'(x)[dx] = (\nabla f)^T dx$ as in our definition of the gradient), and if f maps column vectors to column vectors gives the Jacobian matrix. Otherwise, simply write down f' as a linear operation.

1. $f(x) = \|x\| = \sqrt{x^T x}$ for $x \in \mathbb{R}^m$.
2. $f(x) = \frac{x^T(A+\|x\|^2 I)x}{x^T x}$ for $x \in \mathbb{R}^m$, A being a constant $m \times m$ matrix, and I being the $m \times m$ identity matrix.
3. $f(A) = A^{-2}$ where A is an $m \times m$ matrix.
4. $f(A) = (\operatorname{trace} A)^9$ where A is an $m \times m$ matrix.
5. $f(x) = A(x \mathbin{.*} x)$ where A is an $m \times n$ matrix, $x \in \mathbb{R}^n$, and $.*$ denotes *elementwise* multiplication (also called a Hadamard product) in Julia/Matlab notation.

Solution:

1. By the chain rule, $df = \sqrt{x^T x} = \frac{d(x^T x)}{2\sqrt{x^T x}}$ (using familar single-variable derivative of square root). From class (via the product rule), $d(x^T x) = 2x^T dx$, so this yields:

$$df = f'(x)[dx] = \frac{2x^T dx}{2\sqrt{x^T x}} = \underbrace{\frac{x^T}{\|x\|}}_{f'=(\nabla f)^T} dx$$

giving $\boxed{\nabla f = x/\|x\|}$.

2. We can use the quotient rule $f = g/h \implies df = (h\,dg - g\,dh)/h^2 = (dg - f\,dh)/h$ just as in ordinary single-variable calculus (derivation from the product rule still works: $dg = d(fh) = df\,h + f\,dh$). Here, the denominator $h = x^T x$ so $dh = 2x^T dx$ (from class, by the product rule). And the numerator is $g = x^T(A + \|x\|^2 I)x$. Applying the product rule to this, and using the result from class for $d(x^T A x) = x^T(A + A^T)dx$ along with $\|x\|^2 = x^T x = h \implies d(\|x\|^2) = 2x^T dx$ again, we obtain for the numerator:

$$dg = x^T\left(A + A^T + 2(x^T x)I\right)dx + \underbrace{x^T(2(x^T dx)I)x}_{=2x^T x(x^T dx) = x^T(2x^T x I)dx} = x^T\left(A + A^T + 4(x^T x)I\right)dx\,,$$

where we have used the fact that multiplication by scalars like $x^T dx$ commute with all matrix/vector operations. So, we obtain an overall derivative f' of

$$df = f'(x)[dx] = \frac{x^T\left(A + A^T + 4(x^T x)I\right)dx - f(x)(2x^T)dx}{x^T x} = (\nabla f)^T dx\,,$$

giving

$$\boxed{\nabla f = \frac{\left(A + A^T + 4(x^T x)I\right)x - 2f(x)x}{x^T x}}. \tag{1}$$

(This could be written in a few other ways)

3. By the product rule, $d(A^{-2}) = d(A^{-1})A^{-1} + A^{-1}d(A^{-1})$, and since we know $d(A^{-1}) = -A^{-1}\,dA\,A^{-1}$ from class, we immediately obtain:

$$df = f'(A)[dA] = d(A^{-2}) = \boxed{-A^{-1}\,dA\,A^{-2} - A^{-2}\,dA\,A^{-1}},$$

which is a linear operation on dA.

4. Trace is a linear operation so $d\,\text{trace}\,A = \text{trace}\,dA$. Since the trace is a scalar α, the familiar single-variable power-rule applies $d(\alpha^9) = 9\alpha^8 d\alpha$. Putting these together (chain rule), we obtain:

$$df = f'(A)[dA] = \boxed{9(\text{trace}\,A)^8\,\text{trace}\,dA},$$

which is a linear operation on dA.

(We could turn this into a matrix gradient $\nabla f = 9(\text{trace}\,A)^8 I$ using the Frobenius inner product, but you weren't required to do this.)

5. $f(x) = A(x\,.*\,x)$ where A is an $m \times n$ matrix, $x \in \mathbb{R}^n$, and $.*$ denotes *elementwise* multiplication (also called a Hadamard product) in Julia/Matlab notation.

It is easy to derive an elementwise product rule $d(x\,.*\,x) = (dx\,.*\,x) + (x\,.*\,dx) = 2x\,.*\,dx$ by applying the ordinary product rule to each element and noting that $.*$ is commutative. Then, from the product rule, we have:

$$df = f'(x)[dx] = Ad(x\,.*\,x) = 2A(x\,.*\,dx)\,.$$

But since this is a function that maps column vectors to column vectors, we should be able to write this as a Jacobian matrix multiplying dx. How? Simply think about what these operations to dx: we first multiply each element of dx by the corresponding element of x, and then we multiply by A (the dot product of each row of A by this vector). But that is equivalent to *scaling each column of* A by the corresponding element of x, and then multiplying the resulting matrix by dx in the ordinary way. In Julia's "broadcast" notation, this operation of scaling each column of A is simply $2A\,.*\,x^T$, so we can write

$$df = 2(A\,.*\,x^T)dx$$

and the Jacobian is $\boxed{J = 2A\,.*\,x^T}$. Equivalently, the entries of the Jacobian are $\boxed{J_{ij} = 2A_{ij}x_j}$. (Any clear notation/description is acceptable here.)

Problem 4 (5 points)

Suppose that $f(t) = A(t)$ is a function that maps scalars $t \in \mathbb{R}$ to $m \times n$ matrices $A(t)$. For example, $A(t) = \begin{pmatrix} \sin(t) & 0 & \cos(t) \\ t & t^2 & t^3 \end{pmatrix}$. Explain why $f'(t)[dt]$, following our general definition, must simply correspond to taking the ordinary single-variable calculus derivative of each element of $A(t)$ (the "elementwise" derivative) and multiplying it by the scalar dt. That is, $f'(t) = A'(t)$ is the elementwise derivative.

Solution:

From our general defintion, $df = f'(t)dt = dA = A(t + dt) - A(t)$. But because matrix subtraction is done elementwise, this is a matrix whose entries are $(dA)_{ij} = d(A_{ij}) = A_{ij}(t + dt) - A_{ij}(t) = A'_{ij}(t)dt$. $A_{ij}(t)$ is a single-variable function mapping $\mathbb{R}$ to $\mathbb{R}$, however, so ordinary single-variable calculus applies to the derivative $A'_{ij}(t)$. Hence $f'(t) = A'(t)$ is simply the elementwise derivative.

Problem 5 (3+3+6+3 points)

If you are not familiar with 2d convolution operations (or even if you are), watch (a little of the start of) the YouTube video `https://www.youtube.com/watch?v=yb2tPt0QVPY` that explains them.

1. The (linear) convolution of an $m \times n$ array ("matrix") with the 3×3 Sobel kernel $\begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix} \begin{pmatrix} -1 & 0 & 1 \end{pmatrix} = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}$ results in a ______ by ______ array?
2. Explain why this map from arrays to arrays is a linear operation.
3. Let X be a 2024×2024 array, and Y be the result of convolving X with Sobel. Describe the matrix M that satisfies:

$$\operatorname{vec}(Y) = M \operatorname{vec}(X)\,.$$

   What is the size of M? Express M in terms of Kronecker products of much smaller matrices (hint: 2d convolution with this Sobel kernel is "separable" into 1d convolutions acting on the rows and columns of X, and you can express these as matrices multiplying X on the ______ and ______, respectively).
4. Convolutions like this are very common linear operations, and yet they are not normally implemented by constructing an explicit matrix then multiplying it by a vector (even for 1d convolutions, much less 2d), no matter what you may have learned in linear algebra classes. Why is that?

Solution:

1. As defined in the linked video, a convolution with a 3×3 kernel by default shrinks the image by 1 in each direction from the edges—otherwise, you would need to define a "boundary condition" describing what to do for values beyond the edge of the image. So, the result should be a $\boxed{(m-2) \times (n-2)}$ array.
2. The definition of convolution $Y = f(X)$ is that each element of Y is a linear combination of the entries of X, which is obviously linear. In this particular case, $Y_{ij} = -X_{i-1,j-1} + X_{i-1,j+1} - 2X_{i,j-1} + 2X_{i,j+1} - X_{i+1,j-1} + X_{i+1,j+1}$, which is linear in X.
3. The input is a 2024×2024 array ("matrix") X, and hence from above the output is a 2022×2022 array Y. Hence $\operatorname{vec}(X)$ has 2024^2 entries and $\operatorname{vec}(Y) = M\operatorname{vec}(X)$ has 2022^2 entries, so M must be $\boxed{2022^2 \times 2024^2 = 4088484 \times 4096576}$, a huge matrix.

   More explicitly, we can write this M in terms of Kronecker products of two 2022×2024 matrices, because this Sobel kernel is *separable* into a convolution of each *column* of X with $[1, 2, 1]$ (an "averaging" kernel, without the 1/4 normalization), and of each *row* of X with $[-1, 0, 1]$ (a "difference" kernel). That is, each column is multiplied by the 2022×2024 convolution matrix:

$$C = \begin{pmatrix} 1 & 2 & 1 & & & \\ & 1 & 2 & 1 & & \\ & & 1 & 2 & 1 & \\ & & & \ddots & \ddots & \ddots \end{pmatrix}$$

To multiply *each column* of X by C, we simply do CX. Similarly, to convolve each row with $[-1,0,1]$ corresponds to $XR^T = (RX^T)^T$, i.e. convolving each column of X^T (= rows of X) with the 2022×2024 convolution matrix:

$$R = \begin{pmatrix} -1 & 0 & 1 & & & \\ & -1 & 0 & 1 & & \\ & & -1 & 0 & 1 & \\ & & & \ddots & \ddots & \ddots \end{pmatrix}$$

Putting these together, we have

$$Y = CXR^T \iff \text{vec}(Y) = (R \otimes C)\text{vec}(X)\,,$$

hence $\boxed{M = R \otimes C}$.

4. A reasonably smart way to compute this convolution is to calculate each element of Y by the formula $Y_{ij} = -X_{i-1,j-1} + X_{i-1,j+1} - 2X_{i,j-1} + 2X_{i,j+1} - X_{i+1,j-1} + X_{i+1,j+1}$, which costs 5 additions and 2 multiplications per output. If we let $N = 2022^2$ be the number of outputs, this is $\Theta(N)$ computational cost, with minimal storage ($\Theta(N)$ for the output Y, and only a few numbers for the coefficients of the convolution).

   In contrast, if we formed M explicitly as a dense matrix (i.e. storing all entries, whether they are zero or not) and computed $\text{vec}(Y) = M\text{vec}(X)$, then it would require $\Theta(N^2)$ computational cost and storage, which is *millions of times* more expensive for an image of this size. In fact, just *storing M* as a dense array of 64-bit floating-point values would require $2022^2 \times 2024^2 \times 8 \approx 1.34 \times 10^{14}$ bytes, or over *100 terabytes*, which is ridiculous.

   Of course, there is a middle ground. One could store M in a "sparse matrix" data structure that only stores the nonzero entries. The cost then once again becomes $\Theta(N)$. But the efficiency will still be worse (by a constant factor), because of the complexity of looking up values in such a data structure doesn't exploit the regularity of the convolution operation, versus the "matrix-free" approach of just applying the 3×3 kernel to each pixel.

Problem 6 (3+4+3+3 points)

Let $f(A)$ be a function that maps $m \times m$ matrices to $m \times m$ matrices. Recall that its derivative $f'(A)$ is a linear operator that maps any change δA in A to the corresponding change $\delta f = f(A + \delta A) - f(A) \approx f'(A)[\delta A]$, to first order in δA.

In this problem, you will study and prove a remarkable identity (Mathias, 1996): if $f(A)$ is sufficiently smooth,[1] then for *any* δA (not necessarily small!) the following formula holds:

$$f\left(\underbrace{\begin{bmatrix} A & \delta A \\ & A \end{bmatrix}}_{M}\right) = \begin{bmatrix} f(A) & f'(A)[\delta A] \\ & f(A) \end{bmatrix}.$$

That is, one applies f to a $2m \times 2m$ "block upper-trianguar" matrix M (blank lower-left = zeros), and the desired derivative is in the upper-right $m \times m$ corner of the result $f(M)$.

1. Check this identity numerically in Julia against a finite-difference approximation for $f(A) = \exp(A)$ (the matrix exponential e^A, computed by `exp(A)` in Julia, or `expm` in Scipy or Matlab), for a random 3×3 `A = randn(3,3)` and a random small perturbation `dA = randn(3,3) * 1e-8`; note that you can make the block matrix above by `using LinearAlgebra` followed by `M = [A dA; 0I A]`, and you can extract an upper-right corner by (*e.g.*) `M[1:3,4:6]`.
2. Prove the identity by explicit computation for the cases: $f(A) = I$, $f(A) = A$, $f(A) = A^2$, and $f(A) = A^3$. (Two of these are trivial! This is "bargain-basement induction": do a few small examples and see the pattern.)
3. Prove the identity for $f(A) = A^n$ for any $n \geq 0$ by induction: assume it is works for A^{n-1} and show using the product rule that it therefore must work for A^n. (You already proved the trivial $n = 0$ base case in the previous part.)

   Remark: Once it works for any A^n, it immediately follows that it works for any $f(A)$ described by a Taylor series, such as $\exp(A) = I + A + A^2/2 + A^3/6 + \cdots + A^n/n! + \cdots$, since such a function is just a linear combination of A^n terms.
4. Prove the identity for $f(A) = A^{-1}$ by explicit computation: since we know (from class) that $f'(A)[\delta A] = -A^{-1}\,\delta A\,A^{-1}$, plug this into the right-hand side of the formula above and show that it is the inverse of M: multiply by M and show you get I.

[1] The result is easiest to show when $f(A)$ has a Taylor series (is "analytic"), and in fact you will do this below, but Higham (2008) shows that it remains true whenever f is $2m-1$ times differentiable, or even just differentiable if A is diagonalizable.

Solution:

1. See attached Julia notebook.
2. These three cases are:
   (a) $f(A) = A^0 = I$: in this case $f(M) = I$ ($2m \times 2m$), and so the upper-right block is **zero**. This, of course, is the correct result $d(I) = 0$.
   (b) $f(A) = A$: in this case, $f(M) = M$, and the upper-right block is δA. Again, this is the correct result: $df = f'(A)[dA] = d(A) = dA$, so $f'(A)[\delta A] = \delta A$.
   (c) $f(A) = A^2$. In this case,
   $$f(M) = M^2 = \begin{pmatrix} A & \delta A \\ & A \end{pmatrix} \begin{pmatrix} A & \delta A \\ & A \end{pmatrix} = \begin{pmatrix} A^2 & A\,\delta A + \delta A\, A \\ & A^2 \end{pmatrix}$$
   by the usual "rows-times-columns" rule (which works for matrix *blocks* as well as for scalar elements). But then the upper-right block $A\,\delta A + \delta A\, A$ is precisely $f'(A)[\delta A]$ as derived in class by the product rule.
   (d) $f(A) = A^3$. Building off the previous part, we have
   $$f(M) = M^3 = M^2 M = \begin{pmatrix} A^2 & A^2\,\delta A + \delta A\, A \\ & A \end{pmatrix} \begin{pmatrix} A & \delta A \\ & A \end{pmatrix} = \begin{pmatrix} A^3 & A^2\,\delta A + A\,\delta A\, A + \delta A\, A^2 \\ & A^3 \end{pmatrix}$$
   again by the usual "rows-times-columns" rule. The upper-right block $\delta A + A\,\delta A\, A + \delta A\, A^2$ is again $f'(A)[\delta A]$ as derived in class, corresponding to $d(A^3) = dA + A\,dA\,A + dA\,A^2$
3. For an inductive proof, we assume (for $n > 0$) that the identity holds for $n-1$, i.e. that:
   $$M^{n-1} = \begin{pmatrix} A^{n-1} & (A^{n-1})'[\delta A] \\ & A^{n-1} \end{pmatrix}.$$
   It then follows that
   $$M^n = M^{n-1}M = \begin{pmatrix} A^{n-1} & (A^{n-1})'[\delta A] \\ & A^{n-1} \end{pmatrix} \begin{pmatrix} A & \delta A \\ & A \end{pmatrix} = \begin{pmatrix} A^n & A^{n-1}\,\delta A + (A^{n-1})'[\delta A]\,A \\ & A^n \end{pmatrix},$$
   again by the usual "rows-times-columns" rule. But the upper-right block corresponds exactly to the product rule $d(A^n) = d(A^{n-1}A) = A^{n-1}dA + d(A^{n-1})A$, so it is indeed $(A^n)'[\delta A]$ as desired.

   As noted in the problem, the inductive base case $n = 0$ was already shown in the previous part, so our result must now hold for all $n \geq 0$.
4. For $f(A) = A^{-1}$, we know from class that $f'(A)[\delta A] = -A^{-1}\,\delta A\, A^{-1}$. We now want to show that
   $$M^{-1} = \begin{pmatrix} A^{-1} & -A^{-1}\,\delta A\, A^{-1} \\ & A^{-1} \end{pmatrix},$$
   which we can establish by explicit multiplication with M (on either the left or right):
   $$\begin{pmatrix} A^{-1} & -A^{-1}\,\delta A\, A^{-1} \\ & A^{-1} \end{pmatrix} \begin{pmatrix} A & \delta A \\ & A \end{pmatrix} = \begin{pmatrix} A^{-1}A & A^{-1}\,\delta A - A^{-1}\,\delta A\, A^{-1}A \\ & A^{-1}A \end{pmatrix} = I$$
   as desired: $f(M) = M^{-1}$ indeed must have the correct derivative $-A^{-1}\,\delta A\, A^{-1}$ in the upper-right block and A^{-1} on the diagonal blocks.