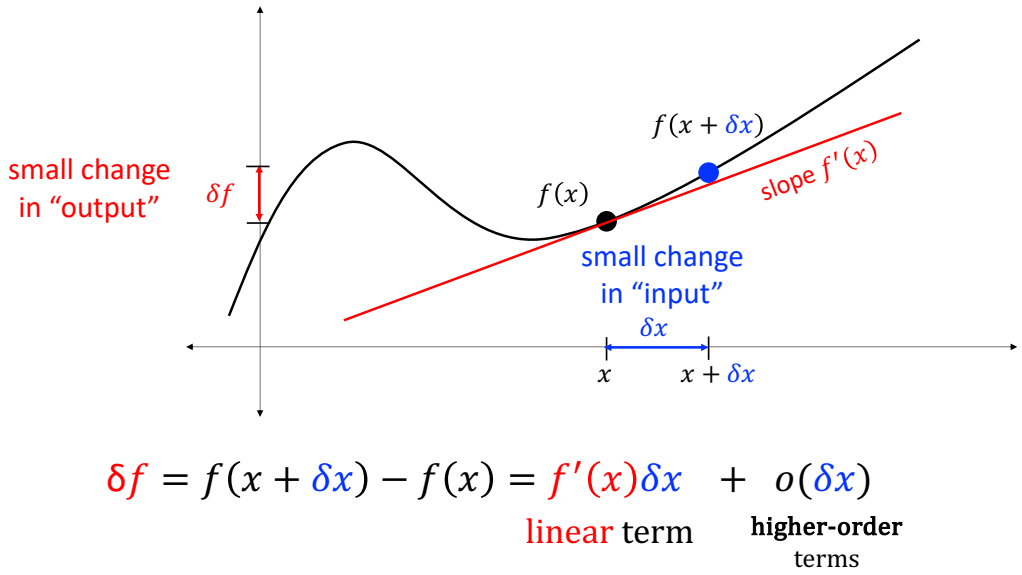

Figure 1: The essence of a derivative is *linearization*: predicting a small change δf in the output $f(x)$ from a small change δx in the input x, to *first order* in δx.

2 Derivatives as Linear Operators

We are now going to revisit the notion of a derivative in a way that we can generalize to higher-order arrays and other vector spaces. We will get into more detail on differentiation as a linear operator, and in particular, will dive deeper into some of the facts we have stated thus far.

2.1 Revisiting single-variable calculus

In a first-semester single-variable calculus course (like 18.01 at MIT), the derivative $f'(x)$ is introduced as the slope of the tangent line at the point $(x, f(x))$, which can also be viewed as a *linear approximation* of f near x. In particular, as depicted in Fig. 1, this is equivalent to a prediction of the *change* δf *in the "output"* of $f(x)$ from a small *change* δx *in the "input"* to first order (*linear*) in δx:

$$\delta f = f(x+\delta x) - f(x) = f'(x)\,\delta x + \underbrace{(\text{higher-order terms})}_{o(\delta x)}.$$

We can more precisely express these higher-order terms using asymptotic "little-o" notation "$o(\delta x)$", which denotes any function whose magnitude shrinks much faster than $|\delta x|$ as $\delta x \to 0$, so that for sufficiently small δx it is negligible compared to the linear $f'(x)\,\delta x$ term. (Variants of this notation are commonly used in computer science, and there is a formal definition that we omit here.[1]) Examples of such higher-order terms include $(\delta x)^2$, $(\delta x)^3$, $(\delta x)^{1.001}$, and $(\delta x)/\log(\delta x)$.

Remark 6. *Here, δx is* not *an infinitesimal but rather a small number. Note that our symbol "δ" (a Greek lowercase "delta") is* not *the same as the symbol "∂" commonly used to denote partial derivatives.*

This notion of a derivative may remind you of the first two terms in a Taylor series $f(x+\delta x) = f(x)+f'(x)\,\delta x+\cdots$ (though in fact it is much more basic than Taylor series!), and the notation will generalize nicely to higher dimensions

[1] Briefly, a function $g(\delta x)$ is $o(\delta x)$ if $\lim_{\delta x\to 0} \frac{\|g(\delta x)\|}{\|\delta x\|} = 0$. We will return to this subject in Section 5.2.

and other vector spaces. In differential notation, we can express the same idea as:

$$df = f(x + dx) - f(x) = f'(x)\, dx.$$

In this notation we implicitly drop the $o(\delta x)$ term that vanishes in the limit as δx becomes infinitesimally small.

We will use this as the more generalized definition of a derivative. In this formulation, we avoid *dividing* by dx, because soon we will allow x (and hence dx) to be something other than a number—if dx is a vector, we won't be *able* to divide by it!

2.2 Linear operators

From the perspective of linear algebra, given a function f, we consider the derivative of f, to be the *linear operator* $f'(x)$ such that

$$df = f(x + dx) - f(x) = f'(x)[dx].$$

As above, you should think of the differential notation dx as representing an *arbitrary small* change in x, where we are implicitly dropping any $o(dx)$ terms, i.e. terms that decay faster than linearly as $dx \to 0$. Often, we will omit the square brackets and write simply $f'(x)dx$ instead of $f'(x)[dx]$, but this should be understood as the linear operator $f'(x)$ *acting on* dx—don't write $dx\, f'(x)$, which will generally be nonsense!

This definition will allow us to extend differentiation to *arbitrary vector spaces* of inputs x and outputs $f(x)$. (More technically, we will require vector spaces with a norm $\|x\|$, called "Banach spaces," in order to precisely define the $o(\delta x)$ terms that are dropped. We will come back to the subject of Banach spaces later.)

Recall 7 (Vector Space)
Loosely, a vector space (over $\mathbb{R}$) is a set of elements in which addition and subtraction between elements is defined, along with multiplication by real scalars. For instance, while it does not make sense to multiply arbitrary vectors in $\mathbb{R}^n$, we can certainly add them together, and we can certainly scale the vectors by a constant factor.

Some examples of vector spaces include:

- $\mathbb{R}^n$, as described in the above. More generally, $\mathbb{R}^{n\times m}$, the space of $n \times m$ matrices with real entries. Notice again that, if $n \neq m$, then multiplication between elements is not defined.
- $C^0(\mathbb{R}^n)$, the set of continuous functions over $\mathbb{R}^n$, with addition defined pointwise.

Recall 8 (Linear Operator)
Recall that a linear operator is a map L from a vector v in vector space V to a vector $L[v]$ (sometimes denoted simply Lv) in some other vector space. Specifically, L is linear if

$$L[v_1 + v_2] = Lv_1 + Lv_2 \quad \text{and} \quad L[\alpha v] = \alpha L[v]$$

for scalars $\alpha \in \mathbb{R}$.

Remark: In this course, f' is a map that takes in an x and spits out a linear operator $f'(x)$ (the **derivative** of f at x). Furthermore, $f'(x)$ is a linear map that takes in an input direction v and gives an output vector $f'(x)[v]$ (which we will later interpret as a directional derivative, see Sec. 2.2.1). When the direction v is an infinitesimal dx, the output $f'(x)[dx] = df$ is the **differential** of f (the corresponding infinitesimal change in f).

Notation 9 (Derivative operators and notations)

There are multiple notations for derivatives in common use, along with multiple related concepts of derivative, differentiation, and differentials. In the table below, we summarize several of these notations, and put boxes around the notations adopted for this course:

name	notations	remark
derivative	$\boxed{f'}$, also $\frac{df}{dx}$, Df, f_x, $\partial_x f$, ...	linear operator $f'(x)$ that maps a small change dx in the input to a small change $df = f'(x)[dx]$ in the output In single-variable calculus, this linear operator can be represented by a *single number*, the "slope," e.g. if $f(x) = \sin(x)$ then $f'(x) = \cos(x)$ is the number that we multiply by dx to get $dy = \cos(x)dx$. In multi-variable calculus, the linear operator $f'(x)$ can be represented by a *matrix*, the **Jacobian** J (see Sec. 3), so that $df = f'(x)[dx] = J\,dx$. But we will see that it is not always convenient to express f' as a matrix, even if we can.
differentiation	$\boxed{'}$, $\frac{d}{dx}$, D, ...	linear operator that maps a function f to its derivative f'
difference	$\boxed{\delta x}$ and $\boxed{\delta f} = f(x+\delta x) - f(x)$	small (but *not* infinitesimal) change in the input x and output f (depending implicitly on x and δx), respectively: an element of a vector space, *not* a linear operator
differential	$\boxed{dx}$ and $\boxed{df} = f(x+dx) - f(x)$	arbitrarily small ("infinitesimal"[a] — we drop higher-order terms) change in the input x and output f, respectively: an element of a vector space, *not* a linear operator
gradient	$\boxed{\nabla f}$	the vector whose inner product $df = \langle \nabla f, dx \rangle$ with a small change dx in the input gives the small change df in the output. The "transpose of the derivative." (See Sec. 2.3.)
partial derivative	$\boxed{\frac{\partial f}{\partial x}}$, f_x, $\partial_x f$	linear operator that maps a small change dx in a *single argument* of a multi-argument function to the corresponding change in output, e.g. for $f(x,y)$ we have $df = \frac{\partial f}{\partial x}[dx] + \frac{\partial f}{\partial y}[dy]$.

[a] Informally, one can think of the vector space of infinitesimals dx as living in the same space as x (understood as a small change in a vector, but still a vector nonetheless). Formally, one can define a distinct "vector space of infinitesimals" in various ways, e.g. as a cotangent space in differential geometry, though we won't go into more detail here.

Some examples of linear operators include

- Multiplication by scalars α, i.e. $Lv = \alpha v$. Also multiplication of column vectors v by matrices A, i.e. $Lv = Av$.
- Some functions like $f(x) = x^2$ are obviously nonlinear. But what about $f(x) = x + 1$? This may *look* linear if you plot it, but it is *not* a linear operation, because $f(2x) = 2x + 1 \neq 2f(x)$—such functions, which are linear *plus a nonzero constant*, are known as *affine*.
- There are also many other examples of linear operations that are not so convenient or easy to write down as matrix–vector products. For example, if A is a 3×3 matrix, then $L[A] = AB + CA$ is a linear operator given 3×3 matrices B, C. The transpose $f(x) = x^T$ of a column vector x is linear, but is not given by any matrix multiplied by x. Or, if we consider vector spaces of *functions*, then the calculus operations of differentiation and integration are linear operators too!

2.2.1 Directional derivatives

There is an equivalent way to interpret this linear-operator viewpoint of a derivative, which you may have seen before in multivariable calculus: as a **directional derivative**.

If we have a function $f(x)$ of arbitrary vectors x, then the directional derivative at x in a direction (vector) v is defined as:

$$\frac{\partial}{\partial \alpha} f(x + \alpha v)\bigg|_{\alpha=0} = \lim_{\delta\alpha \to 0} \frac{f(x + \delta\alpha\, v) - f(x)}{\delta\alpha} \tag{1}$$

where α is a *scalar*. This transforms derivatives back into *single-variable calculus* from arbitrary vector spaces. It measures the rate of change of f in the direction v from x. But it turns out that this has a very simple relationship to our linear operator $f'(x)$ from above, because (dropping higher-order terms due to the limit $\delta\alpha \to 0$):

$$f(x + \underbrace{d\alpha\, v}_{dx}) - f(x) = f'(x)[dx] = d\alpha\, f'(x)[v]\,,$$

where we have factored out the scalar $d\alpha$ in the last step thanks to $f'(x)$ being a *linear* operator. Comparing with above, we immediately find that the directional derivative is:

$$\boxed{\frac{\partial}{\partial \alpha} f(x + \alpha v)\bigg|_{\alpha=0} = f'(x)[v]}\,. \tag{2}$$

It is *exactly equivalent* to our $f'(x)$ from before! (We can also see this as an instance of the chain rule from Sec. 2.5.) One lesson from this viewpoint is that it is perfectly reasonable to input an arbitrary *non-infinitesimal* vector v into $f'(x)[v]$: the result is not a df, but is simply a directional derivative.

2.3 Revisiting multivariable calculus, Part 1: Scalar-valued functions

Let f be a scalar-valued function, which takes in "column" vectors $x \in \mathbb{R}^n$ and produces a scalar (in $\mathbb{R}$). Then,

$$df = f(x + dx) - f(x) = f'(x)[dx] = \text{scalar}.$$

Therefore, since dx is a column vector (in an arbitrary direction, representing an arbitrary small change in x), the linear operator $f'(x)$ that produces a scalar df must be a **row vector** (a "1-row matrix", or more formally something called a *covector* or "dual" vector or "linear form")! We call this row vector the *transpose of the gradient*

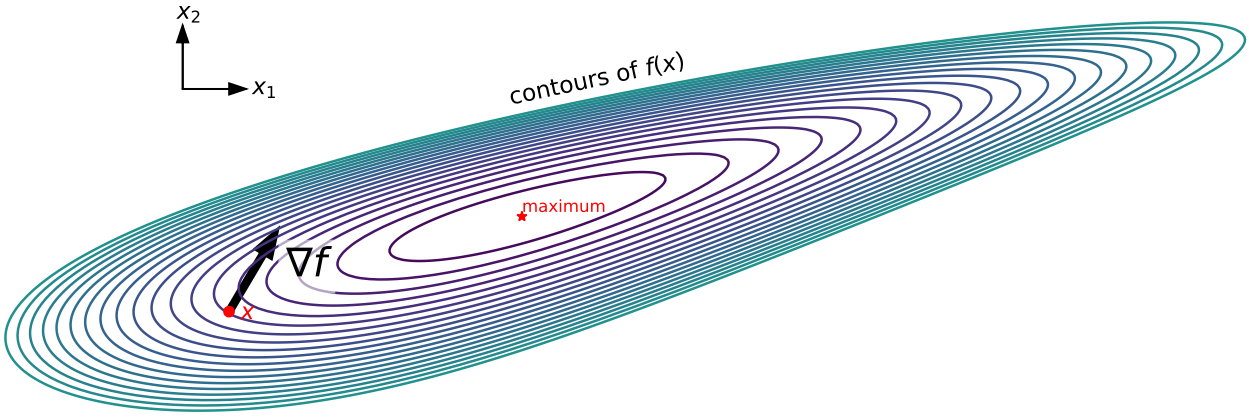

Figure 2: For a real-valued $f(x)$, the gradient ∇f is defined so that it corresponds to the "uphill" direction at a point x, which is perpendicular to the contours of f. Although this may not point exactly towards the nearest local maximum of f (unless the contours are circular), "going uphill" is nevertheless the starting point for many computational-optimization algorithms to search for a maximum.

$(\nabla f)^T$, so that df is the *dot ("inner") product of dx with the gradient*. So we have that

$$df = \nabla f \cdot dx = \underbrace{(\nabla f)^T}_{f'(x)} dx \quad \text{where} \quad dx = \begin{pmatrix} dx_1 \\ dx_2 \\ \vdots \\ dx_n. \end{pmatrix}.$$

Some authors view the gradient as a row vector (equating it with f' or the Jacobian), but treating it as a "column vector" (the transpose of f'), as we do in this course, is a common and useful choice. As a column vector, the gradient can be viewed as the "uphill" (*steepest-ascent*) direction in the x space, as depicted in Fig. 2. Furthermore, it is also easier to generalize to scalar functions of other vector spaces. In any case, for this class, we will *always* define ∇f to *have the same "shape" as x*, so that df is a dot product ("inner product") of dx with the gradient.

This is perfectly consistent with the viewpoint of the gradient that you may remember from multivariable calculus, in which the gradient was a vector of components

$$\nabla f = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{pmatrix};$$

or, equivalently,

$$df = f(x + dx) - f(x) = \nabla f \cdot dx = \frac{\partial f}{\partial x_1} dx_1 + \frac{\partial f}{\partial x_2} dx_2 + \cdots + \frac{\partial f}{\partial x_n} dx_n \,.$$

While a component-wise viewpoint may sometimes be convenient, we want to encourage you to view the vector x as a *whole*, not simply a collection of components, and to learn that it is often more convenient and elegant to differentiate expressions *without* taking the derivative component-by-component, a new approach that will generalize better to more complicated inputs/output vector spaces.

Let's look at an example to see how we compute this differential.

Example 10

Consider $f(x) = x^T Ax$ where $x \in \mathbb{R}^n$ and A is a square $n \times n$ matrix, and thus $f(x) \in \mathbb{R}$. Compute df, $f'(x)$, and ∇f.

We can do this directly from the definition.

$$\begin{aligned} df &= f(x+dx) - f(x) \\ &= (x+dx)^T A(x+dx) - x^T Ax \\ &= \cancel{x^T Ax} + dx^T\, Ax + x^T Adx + \cancel{dx^T Adx}^{\text{higher order}} - \cancel{x^T Ax} \\ &= \underbrace{x^T(A+A^T)}_{f'(x)=(\nabla f)^T} dx \implies \nabla f = (A+A^T)x\,. \end{aligned}$$

Here, we dropped terms with more than one dx factor as these are asymptotically negligible. Another trick was to combine $dx^T Ax$ and $x^T Adx$ by realizing that these are *scalars* and hence equal to their own transpose: $dx^T Ax = (dx^T Ax)^T = x^T A^T dx$. Hence, we have found that $f'(x) = x^T(A + A^T) = (\nabla f)^T$, or equivalently $\nabla f = [x^T(A + A^T)]^T = (A + A^T)x$.

It is, of course, also possible to compute the same gradient component-by-component, the way you probably learned to do in multivariable calculus. First, you would need to write $f(x)$ explicitly in terms of the components of x, as $f(x) = x^T Ax = \sum_{i,j} x_i A_{i,j} x_j$. Then, you would compute $\partial f/\partial x_k$ for each k, taking care that x appears twice in the f summation. However, this approach is awkward, error-prone, labor-intensive, and quickly becomes worse as we move on to more complicated functions. It is much better, we feel, to get used to treating vectors and matrices *as a whole*, not as mere collections of numbers.

2.4 Revisiting multivariable calculus, Part 2: Vector-valued functions

Next time, we will revisit multi-variable calculus (18.02 at MIT) again in a Part 2, where now f will be a vector-valued function, taking in vectors $x \in \mathbb{R}^n$ and giving vector outputs $f(x) \in \mathbb{R}^m$. Then, df will be a m-component column vector, dx will be an n-component column vector, and we must get a linear operator $f'(x)$ satisfying

$$\underbrace{df}_{m \text{ components}} = \underbrace{f'(x)}_{m \times n} \underbrace{dx}_{n \text{ components}},$$

so $f'(x)$ must be an $m \times n$ *matrix* called the *Jacobian* of f!

The Jacobian matrix J represents the linear operator that takes dx to df:

$$df = J dx\,.$$

The matrix J has entries $J_{ij} = \frac{\partial f_i}{\partial x_j}$ (corresponding to the i-th row and the j-th column of J).

So now, suppose that $f : \mathbb{R}^2 \to \mathbb{R}^2$. Let's understand how we would compute the differential of f:

$$df = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} \end{pmatrix} \begin{pmatrix} dx_1 \\ dx_2 \end{pmatrix} = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} dx_1 + \frac{\partial f_1}{\partial x_2} dx_2 \\ \frac{\partial f_2}{\partial x_1} dx_1 + \frac{\partial f_2}{\partial x_2} dx_2 \end{pmatrix}.$$

Let's compute an example.

Example 11

Consider the function $f(x) = Ax$ where A is a constant $m \times n$ matrix. Then, by applying the distributive law for matrix–vector products, we have

$$\begin{aligned} df &= f(x + dx) - f(x) = A(x + dx) - Ax \\ &= \cancel{Ax} + Adx - \cancel{Ax} = Adx = f'(x)dx. \end{aligned}$$

Therefore, $f'(x) = A$.

Notice then that the linear operator A is its own Jacobian matrix!
Let's now consider some derivative rules.

- **Sum Rule**: Given $f(x) = g(x) + h(x)$, we get that

$$df = dg + dh \implies f'(x)dx = g'(x)dx + h'(x)dx.$$

  Hence, $f' = g' + h'$ as we should expect. This is the linear operator $f'(x)[v] = g'(x)[v] + h'(x)[v]$, and note that we can sum linear operators (like g' and h') just like we can sum matrices! In this way, linear operators form a vector space.

- **Product Rule**: Suppose $f(x) = g(x)h(x)$. Then,

$$\begin{aligned} df &= f(x + dx) - f(x) \\ &= g(x + dx)h(x + dx) - g(x)h(x) \\ &= (g(x) + \underbrace{g'(x)dx}_{dg})(h(x) + \underbrace{h'(x)dx}_{dh}) - g(x)h(x) \\ &= gh + dg\, h + g\, dh + \cancel{dg\, dh}^{0} - gh \\ &= dg\, h + g\, dh\,, \end{aligned}$$

  where the $dg\, dh$ term is higher-order and hence dropped in infinitesimal notation. Note, as usual, that dg and h may not commute now as they may no longer be scalars!

Let's look at some short examples of how we can apply the product rule nicely.

Example 12

Let $f(x) = Ax$ (mapping $\mathbb{R}^n \to \mathbb{R}^m$) where A is a constant $m \times n$ matrix. Then,

$$df = d(Ax) = \cancel{dA}^{0}x + Adx = Adx \implies f'(x) = A.$$

We have $dA = 0$ here because A does not change when we change x.

Example 13

Let $f(x) = x^T Ax$ (mapping $\mathbb{R}^n \to \mathbb{R}$). Then,

$$df = dx^T(Ax) + x^T d(Ax) = \underbrace{dx^T Ax}_{= x^T A^T dx} + x^T A dx = x^T(A + A^T)dx = (\nabla f)^T dx\,,$$

and hence $f'(x) = x^T(A + A^T)$. (In the common case where A is symmetric, this simplifies to $f'(x) = 2x^T A$.) Note that here we have applied Example 12 in computing $d(Ax) = Adx$. Furthermore, f is a scalar valued function, so we may also obtain the gradient $\nabla f = (A + A^T)x$ as before (which simplifies to $2Ax$ if A is symmetric).

Example 14 (Elementwise Products)

Given $x, y \in \mathbb{R}^m$, define

$$x .* y = \begin{pmatrix} x_1 y_1 \\ \vdots \\ x_m y_m \end{pmatrix} = \underbrace{\begin{pmatrix} x_1 & & & \\ & x_2 & & \\ & & \ddots & \\ & & & x_m \end{pmatrix}}_{\mathrm{diag}(x)} \begin{pmatrix} y_1 \\ \vdots \\ y_m \end{pmatrix},$$

the element-wise product of vectors (also called the Hadamard product), where for convenience below we also define $\mathrm{diag}(x)$ as the $m \times m$ diagonal matrix with x on the diagonal. Then, given $A \in \mathbb{R}^{m,n}$, define $f : \mathbb{R}^n \to \mathbb{R}^m$ via

$$f(x) = A(x .* x).$$

As an exercise, one can verify the following:

(a) $x .* y = y .* x$,

(b) $A(x .* y) = A\,\mathrm{diag}(x)\,y$.

(c) $d(x .* y) = (dx) .* y + x .* (dy)$. So if we take y to be a constant and define $g(x) = y .* x$, its Jacobian matrix is $\mathrm{diag}(y)$.

(d) $df = A(2x .* dx) = 2A\,\mathrm{diag}(x)\,dx = f'(x)[dx]$, so the Jacobian matrix is $J = 2A\,\mathrm{diag}(x)$.

(e) Notice that the directional derivative (Sec. 2.2.1) of f at x in the direction v is simply given by $f'(x)[v] = 2A(x .* v)$. One could also check numerically for some arbitrary A, x, v that $f(x + 10^{-8}v) - f(x) \approx 10^{-8}(2A(x .* v))$.

2.5 The Chain Rule

One of the most important rules from differential calculus is the chain rule, because it allows us to differentiate complicated functions built out of compositions of simpler functions. This chain rule can also be generalized to our differential notation in order to work for functions on arbitrary vector spaces:

- **Chain Rule**: Let $f(x) = g(h(x))$. Then,

$$\begin{aligned} df = f(x+dx) - f(x) &= g(h(x+dx)) - g(h(x)) \\ &= g'(h(x))[h(x+dx) - h(x)] \\ &= g'(h(x))[h'(x)[dx]] \\ &= g'(h(x))h'(x)[dx] \end{aligned}$$

  where $g'(h(x))h'(x)$ is a composition of g' and h' as matrices.

  In other words, $f'(x) = g'(h(x))h'(x)$: the Jacobian (linear operator) f' is simply the *product (composition) of the Jacobians*, $g'h'$. Ordering matters because linear operators do not generally commute: left-to-right = outputs-to-inputs.

Let's look more carefully at the *shapes* of these Jacobian matrices in an example where each function maps a column vector to a column vector:

Example 15

Let $x \in \mathbb{R}^n$, $h(x) \in \mathbb{R}^p$, and $g(h(x)) \in \mathbb{R}^m$. Then, let $f(x) = g(h(x))$ mapping from $\mathbb{R}^n$ to $\mathbb{R}^m$. The chain rule then states that

$$f'(x) = g'(h(x))h'(x),$$

which makes sense as g' is an $m \times p$ matrix and h' is a $p \times n$ matrix, so that the product gives an $m \times n$ matrix f'! However, notice that this is *not* the same as $h'(x)g'(h(x))$ as you cannot (if $n \neq m$) multiply a $p \times n$ and an $m \times p$ matrix together, and even if $n = m$ you will get the wrong answer since they probably won't commute.

Not only does the order of the multiplication matter, but the associativity of matrix multiplication matters *practically*. Let's consider a function

$$f(x) = a(b(c(x)))$$

where $c : \mathbb{R}^n \to \mathbb{R}^p$, $b : \mathbb{R}^p \to \mathbb{R}^q$, and $a : \mathbb{R}^q \to \mathbb{R}^m$. Then, we have that, by the chain rule,

$$f'(x) = a'(b(c(x)))b'(c(x))c'(x).$$

Notice that this is the same as

$$f' = (a'b')c' = a'(b'c')$$

by associativity (omitting the function arguments for brevity). The left-hand side is multiplication from left to right, and the right-hand side is multiplication from right to left.

But who cares? Well it turns out that associativity is deeply important. So important that the two orderings have names: multiplying left-to-right is called "reverse mode" and multiplying right-to-left is called "forward mode" in the field of *automatic differentiation* (AD). Reverse-mode differentation is also known as an "adjoint method" or "backpropagation" in some contexts, which we will explore in more detail later. Why does this matter? Let's think about the computational cost of matrix multiplication.

2.5.1 Cost of Matrix Multiplication

If you multiply a $m \times q$ matrix by a $q \times p$ matrix, you normally do it by computing mp dot products of length q (or some equivalent re-ordering of these operations). To do a dot product of length q requires q multiplications and $q - 1$ additions of scalars. Overall, this is approximately $2mpq$ scalar operations in total. In computer science, you would write that this is "$\Theta(mpq)$": the computational effort is *asymptotically proportional* to mpq for large m, p, q.

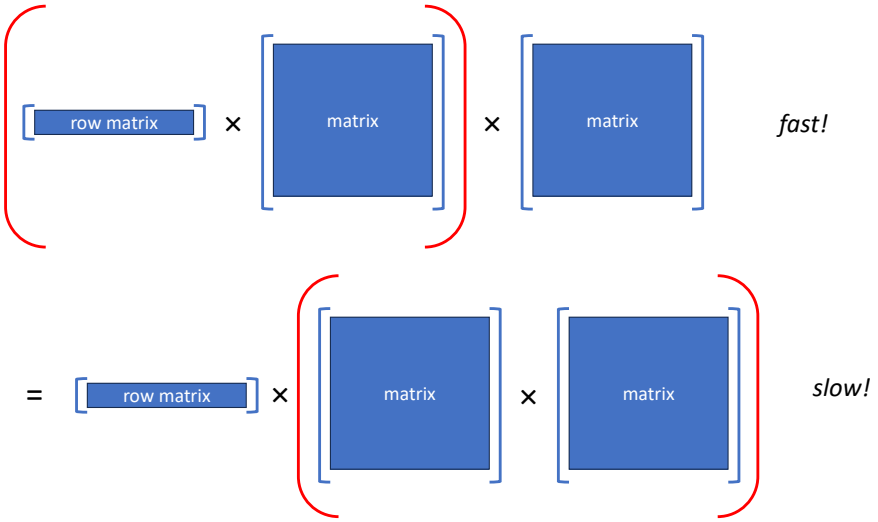

Figure 3: Matrix multiplication is *associative*—that is, $(AB)C = A(BC)$ for all A, B, C—but multiplying left-to-right can be much more efficient than right-to-left if the leftmost matrix has only one (or few) rows, as shown here. Correspondingly, the order in which you carry out the chain rule has dramatic consequences for the computational effort required. Left-to-right is known as "reverse mode" or "backpropagation", and is best suited to situations where there are many fewer outputs than inputs.

So why does the order of the chain rule matter? Consider the following two examples.

Example 16

Suppose you have a lot of inputs $n \gg 1$, and only one output $m = 1$, with lots of intermediate values, i.e. $q = p = n$. Then reverse mode (left-to-right) will cost $\Theta(n^2)$ scalar operations while forward mode (right-to-left) would cost $\Theta(n^3)$! This is a huge cost difference, depicted schematically in Fig. 3.

Conversely, suppose you have a lot of outputs $m \gg 1$ and only one input $n = 1$, with lots of intermediate values $q = p = m$. Then reverse mode would cost $\Theta(m^3)$ operations but forward mode would be only $\Theta(m^2)$!

Moral: If you have a lot of inputs and few outputs (the usual case in machine learning and optimization), compute the chain rule left-to-right (reverse mode). If you have a lot of outputs and few inputs, compute the chain rule right-to-left (forward mode). We return to this in Sec. 8.4.

2.6 Beyond Multi-Variable Derivatives

Now let's compute some derivatives that go beyond first-year calculus, where the inputs and outputs are in more general vector spaces. For instance, consider the following examples:

Example 17

Let A be an $n \times n$ matrix. You could have the following matrix-valued functions. For example:

- $f(A) = A^3$,
- $f(A) = A^{-1}$ if A is invertible,
- or U, where U is the resulting matrix after applying Gaussian elimination to A!

You could also have scalar outputs. For example:

- $f(A) = \det A$,
- $f(A) = \text{trace } A$,
- or $f(A) = \sigma_1(A)$, the largest singular value of A.

Let's focus on two simpler examples for this lecture.

Example 18

Let $f(A) = A^3$ where A is a square matrix. Compute df.

Here, we apply the chain rule one step at a time:

$$df = dA\, A^2 + A\, dA\, A + A^2\, dA = f'(A)[dA].$$

Notice that this is not equal to $3A^2$ (unless dA and A commute, which won't generally be true since dA represents an *arbitrary* small change in A). The right-hand side is a linear operator $f'(A)$ acting on dA, but it is not so easy to interpret it as simply a single "Jacobian" matrix multiplying dA!

Example 19

Let $f(A) = A^{-1}$ where A is a square invertible matrix. Compute $df = d(A^{-1})$.

Here, we use a slight trick. Notice that $AA^{-1} = \mathrm{I}$, the identity matrix. Thus, we can compute the differential using the product rule (noting that $d\,\mathrm{I} = 0$, since changing A does not change I) so

$$d(AA^{-1}) = dA\, A^{-1} + A\, d(A^{-1}) = d(\mathrm{I}) = 0 \implies d(A^{-1}) = -A^{-1}\, dA\, A^{-1}.$$

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