n-dimensional differentiation – exercices

Joans Groschwitz Antoine Venant

Reminder and notational remarks

- When it exits derivative of a function $f : \mathbb{R} \mapsto \mathbb{R}$ is defined **at a specific point** a:

$$\frac{\partial f}{\partial x}(a) = f'(a) = \overbrace{\lim_{h \to 0} \frac{f(a+h) - f(a)}{h}}^{\text{depends on } a!}.$$

- Derived function f' is – as the name gives away – a function: $f' : x \mapsto f'(x)$ which to a point associates the derivative at this point. $\frac{\partial f}{\partial x}$ also is a function. For instance, if $f : x \mapsto x^2$ then $\frac{\partial f}{\partial x}(a) = 2a$. ⚠For simplicity one generally writes $\frac{\partial f}{\partial x} = 2x$ instead of $\frac{\partial f}{\partial x}(x) = 2x$. Note the two distinct meanings of x here: as a **variable index** and has a **real value**.

⚠ the partial derivative notation is somewhat misleading: it contextually builds on a specific variable naming choice which is mathematically insignificant: $f : x \mapsto x^2$ and $g : y \mapsto y^2$ denote the exact same function ; so f' and g' should also denote the exact same derived function, and they do. But $\frac{\partial g}{\partial x}$ is not meaningful in this context whereas $\frac{\delta f}{\delta x}$ is.

- For any fonction $f : \left\{ \begin{array}{c} \mathbb{R}^n \mapsto \mathbb{R}^m \\ \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \mapsto \begin{pmatrix} f_1(x_1, \ldots, x_n) \\ \vdots \\ f_m(x_1, \ldots, x_n) \end{pmatrix} \end{array} \right.$ and any point $A = \begin{pmatrix} a_1 \\ \vdots \\ a_n \end{pmatrix}$ and any $i \in [1, n]$ one can define a function

$$f \upharpoonright a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_n : \begin{array}{c} \mathbb{R} \mapsto \mathbb{R} \\ x_i \mapsto f(a_1, \ldots, a_{i-1}, x_i, a_{i+1}, \ldots, a_n) \end{array}.$$

  By definition:

$$\frac{\partial f}{\partial x_i}(A) = (f \upharpoonright a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_n)'(a_i) = \frac{\partial f \upharpoonright a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_n}{\partial x_i}(a_i).$$

Again, $\frac{\partial f}{\partial x_i}$ is **a function**, but one also overloads it to denote $\frac{\partial f}{\partial x_i}(x_1, \ldots, x_n)$ – again not the two distinct meanings of x_i.

- Consider $f : \left\{ \begin{array}{ccc} \mathbb{R}^n & \mapsto & \mathbb{R}^m \\ \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} & \mapsto & \begin{pmatrix} f_1(x_1, \ldots, x_n) \\ \vdots \\ f_m(x_1, \ldots, x_n) \end{pmatrix} \end{array} \right.$. When it exists, the jacobian is also defined **at a point** $A = \begin{pmatrix} a_1 \\ \vdots \\ a_n \end{pmatrix}$.

$$J_f(A) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1}(A) & \ldots & \frac{\partial f_1}{\partial x_n}(A) \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1}(A) & \ldots & \frac{\partial f_m}{\partial x_n}(A) \end{pmatrix}.$$

Again, J_f is both used to denote a (matrix of) **function**(s), but one also (contextually) writes J_f as a shorthand for $J_f(x_1, \ldots, x_n)$, which is fully compatible with previous conventions:

$$J_f = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \ldots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \ldots & \frac{\partial f_m}{\partial x_n} \end{pmatrix}.$$

- The j^{th} column of $J_f(A)$ is the vector $\begin{pmatrix} \frac{\partial f_1}{\partial x_j}(A) \\ \vdots \\ \frac{\partial f_m}{\partial x_j}(A) \end{pmatrix}$. As a shorthand for it, one writes $\frac{\partial f}{\partial x_j}(A)$ (and again simply $\frac{\partial f}{\partial x_j}$ when implicitely $A = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$).

- The j^{th} line of $J_f(A)$ is the vector $(\frac{\partial f_i}{\partial x_1}(A), \ldots, \frac{\partial f_i}{\partial x_n}(A)) = \nabla f_i(A)$ is the gradient of f_i at point A.

- When considered as matrices, vectors can be seen either as single column or single row matrices on the other hand. More formally, there is an isomorphism between $\mathbb{R}^n$ and (column) $(n, 1)$ matrices on one hand, as well as between $\mathbb{R}^n$ and (row) $(1, n)$ matrices. When using a vector as a matrix without precising, we'll generally mean to use the first, *i.e.* a $(1, n)$ single-column matrix. Beware that, for practical reason, we'll keep using both horizontal $(x_1, \ldots, x_n)$ vertical $\begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$ displays of vectors, depending

on the typographic context. So beware than unless otherwise explicitely stated, a **vector** use as a **matrix** is meant to denote a **single column** matrix.

Exercise 0: looking 'inside' the chain rule.

Assume $f : \mathbb{R}^l \mapsto \mathbb{R}^m$ and $g : \mathbb{R}^k \mapsto \mathbb{R}^l$ and $X \in \mathbb{R}^k$ s.t. f differentiable in $g(X)$ and g differentiable in X.

We remind that the chain rule has:

$$J_{f \circ g}(X) = J_f(g(X)) \times J_g(X).$$

For $i, j \in [1, m] \times [1, k]$, express $\frac{\partial f_i \circ g}{\partial x_j}(X)$ using $\nabla f_i(g(X))$ and $\frac{\partial g}{\partial x_j}(X)$.

Exercise 1

1. Let $M = (m_{i,j})_{i \in [1,l], j \in [1,k]} = \begin{pmatrix} m_{1,1} & \dots & m_{1,k} \\ \vdots & & \vdots \\ m_{l,1} & \dots & m_{l,k} \end{pmatrix}$ be an l, k-matrix with real coefficients (*i.e.* a matrix of real numbers with l lines and k columns). One considers the linear map: $z : \begin{cases} \mathbb{R}^k \mapsto \mathbb{R}^l \\ X \mapsto M \times X \end{cases}$. Compute J_z.

Exercise 2

Compute the Jacobian of the $3D$ relu activation function $R : (x, y, z) \mapsto (x \times \mathbb{1}_{x>0}, y \times \mathbb{1}_{y>0}, z \times \mathbb{1}_{z>0})$ in any point where it is differentiable.

Excercise 3

We consider the following (neural network) function f, defined as $f(P, Q, E) = S(Q \times R(P \times E))$ where:

- $E \in \mathbb{R}^2$ (input to the neural network).
- $P = \begin{pmatrix} p_{1,1} & p_{1,2} \\ p_{2,1} & p_{2,2} \\ p_{3,1} & p_{3,2} \end{pmatrix}$ is a $2, 3$ matrix (weights of the first layer).
- $Q = \begin{pmatrix} q_{1,1} & q_{1,2} & q_{1,3} \\ q_{2,1} & p_{2,2} & q_{2,3} \end{pmatrix}$ is a $3, 2$ matrix (weights of the second layer).
- R is the relu function.
- $S(x, y) = (S_1(x, y), S_2(x, y)) = (\frac{e^x}{e^x + e^y}, \frac{e^y}{e^x + e^y})$ is the softmax normalizer.

f as written above is a function taking the matrices P and Q and the two dimensional vector E to a two dimensional vector, but it can equivalently be seen as a function $\mathbb{R}^{12} \times \mathbb{R}^2 \mapsto \mathbb{R}^2$, where we gather the two matrices components (*i.e.*, the parameters of the neural network) into one single 12 dimensional vector: $\Theta = (\underbrace{p_{1,1}, \ldots, p_{3,2}}_{\text{describes } P}, \underbrace{q_{1,1}, \ldots, q_{2,3}}_{\text{describes } Q})$.

For every fixed value of E, we now have a function $f_E : \Theta \mapsto f_E(\Theta) = f(\Theta, E)$. We first want to compute J_{f_E}. To this aim we'll use several application of the chain rule. Let us write $z_E(\Theta) = P \times E$, $g_E(\Theta) = R(z_E(\Theta))$, and $z'_E(\Theta) = Q \times g_E(\Theta)$.

1. Show that for any variable index $x \in \{p_{1,1}, \ldots, p_{3,2}, q_{1,1}, \ldots, q_{2,3}\}$, $\frac{\partial f_E}{\partial x}(\Theta) = J_S(z'_E(\Theta)) \times \frac{\partial z'_E}{\partial x}(\Theta)$. .
2. Show that $\frac{\partial z'_E}{\partial q_{i,j}}(g_E(\Theta)) = g_E(\Theta)_j \times \begin{pmatrix} \mathbb{1}_{i=1} \\ \mathbb{1}_{i=2} \end{pmatrix}$.
3. Show that $\frac{\partial z'_E}{\partial p_{i,j}}(\Theta) = Q \times \frac{\partial g_E}{\partial p_{i,j}}(\Theta)$
4. Show that $\frac{\partial g_E}{\partial p_{i,j}}(\Theta) = J_R(z_E(\Theta)) \times \frac{\partial z_E}{\partial p_{i,j}}(\Theta)$.
5. Show that $\frac{\partial z_E}{\partial p_{i,j}} = e_j \times \begin{pmatrix} \mathbb{1}_{i=1} \\ \mathbb{1}_{i=2} \\ \mathbb{1}_{i=3} \end{pmatrix}$, where $E = \begin{pmatrix} e_1 \\ e_2 \end{pmatrix}$.
6. We admit that $J_S = \begin{pmatrix} S_1(1-S_1) & -S_1S_2 \\ -S_1S_2 & S_2(1-S_2) \end{pmatrix}$. We let $E_0 = \begin{pmatrix} 1 \\ -1 \end{pmatrix}, P_0 = \begin{pmatrix} 3, 1 \\ 1, 0 \\ -1, 2 \end{pmatrix}$ and $Q_0 = \begin{pmatrix} 1, -1, 3 \\ -2, 5, 4 \end{pmatrix}$, hence the joint 'flat' vectorial representation of P_0 and Q_0 is $\Theta_0 = (3, 1, 1, 0, -1, 2, 1, -1, 3, -2, 5, 4)$. Compute the Jacobian $J_{f_{E_0}}(\Theta_0)$ of f_{E_0} at point Θ_0.
7. We now consider performing one step of gradient descent using a single positive training instance $\langle E_0, + \rangle$ and negative log-likelihood loss. The loss function at a given point Θ is thus $l(\Theta) = -ln(f_{E_0}[1](\Theta))$ (where $f_{E_0}[1] = f_{E_0\,1}$ is the function computing only the first component of f_{E_0}). We execute the step from point Θ_0. Compute the direction of the update: $\nabla l(\Theta_0)$.