Vector Calculus

Gradient

For a scalar valued function f with vector valued input $\mathbf{x} \in \mathbb{R}^n$, the derivative is this row vector of shape $1 \times n$.

$$\frac{df}{d\mathbf{x}} = \nabla_{\mathbf{x}} f = \left[\frac{\partial f}{\partial x_1} \; \frac{\partial f}{\partial x_2} \; \dots \; \frac{\partial f}{\partial x_n} \right]$$

This is the vector along with rate of increase of f (wrt norm of displacement) is highest.

Jacobian for vectors

For a vector valued function $\mathbf{f}(\mathbf{x}) \in \mathbb{R}^m$ with vector valued input $\mathbf{x} \in \mathbb{R}^n$, the derivative is this matrix of shape $m \times n$ called the Jacobian matrix, or simply the Jacobian :

$$\frac{d\mathbf{f}}{d\mathbf{x}} = \nabla_{\mathbf{x}} \mathbf{f} = \mathbf{J} \quad \text{where} \;\; J_{ij} = \frac{\partial f_i}{\partial x_j}$$

Jacobian for Matrices

For a matrix valued function $\mathbf{f}(\mathbf{x}) \in \mathbb{R}^{p \times q}$ with matrix valued $\mathbf{x} \in \mathbb{R}^{m \times n}$, the derivative is again the Jacobian $\mathbf{J}$ which is now a tensor of shape $p \times q \times m \times n$, which can't be written on paper. $\mathbf{J}$ is given by its elements instead as :

$$J_{ijkl} = \frac{\partial f_{ij}}{\partial x_{kl}}$$

But this approach is impractical and what is usually done is that the matrices $\mathbf{f}, \mathbf{x}$ are flattened out into vectors of size pq and mn , and then the Jacobian matrix with respect to these vectors is calculated.

Useful identities for calculating Jacobians

$$\frac{\partial}{\partial \boldsymbol{X}} \boldsymbol{f}(\boldsymbol{X})^\top = \left(\frac{\partial \boldsymbol{f}(\boldsymbol{X})}{\partial \boldsymbol{X}}\right)^\top$$

$$\frac{\partial}{\partial \boldsymbol{X}} \mathrm{tr}(\boldsymbol{f}(\boldsymbol{X})) = \mathrm{tr}\left(\frac{\partial \boldsymbol{f}(\boldsymbol{X})}{\partial \boldsymbol{X}}\right)$$

$$\frac{\partial}{\partial \boldsymbol{X}} \det(\boldsymbol{f}(\boldsymbol{X})) = \det(\boldsymbol{f}(\boldsymbol{X}))\mathrm{tr}\left(\boldsymbol{f}(\boldsymbol{X})^{-1}\frac{\partial \boldsymbol{f}(\boldsymbol{X})}{\partial \boldsymbol{X}}\right)$$

$$\frac{\partial}{\partial \boldsymbol{X}} \boldsymbol{f}(\boldsymbol{X})^{-1} = -\boldsymbol{f}(\boldsymbol{X})^{-1}\frac{\partial \boldsymbol{f}(\boldsymbol{X})}{\partial \boldsymbol{X}}\boldsymbol{f}(\boldsymbol{X})^{-1}$$

$$\frac{\partial \boldsymbol{a}^\top \boldsymbol{X}^{-1} \boldsymbol{b}}{\partial \boldsymbol{X}} = -(\boldsymbol{X}^{-1})^\top \boldsymbol{a}\boldsymbol{b}^\top (\boldsymbol{X}^{-1})^\top$$

$$\frac{\partial \boldsymbol{x}^\top \boldsymbol{a}}{\partial \boldsymbol{x}} = \boldsymbol{a}^\top$$

$$\frac{\partial \boldsymbol{a}^\top \boldsymbol{x}}{\partial \boldsymbol{x}} = \boldsymbol{a}^\top$$

$$\frac{\partial \boldsymbol{a}^\top \boldsymbol{X} \boldsymbol{b}}{\partial \boldsymbol{X}} = \boldsymbol{a}\boldsymbol{b}^\top$$

$$\frac{\partial \boldsymbol{x}^\top \boldsymbol{B} \boldsymbol{x}}{} = \boldsymbol{x}^\top(\boldsymbol{B} + \boldsymbol{B}^\top)$$

$$\frac{\quad}{\partial \boldsymbol{x}} = \boldsymbol{x}^\top(\boldsymbol{B}+\boldsymbol{B}^\top)$$

$$\frac{\partial}{\partial \boldsymbol{s}}(\boldsymbol{x}-\boldsymbol{A}\boldsymbol{s})^\top \boldsymbol{W}(\boldsymbol{x}-\boldsymbol{A}\boldsymbol{s}) = -2(\boldsymbol{x}-\boldsymbol{A}\boldsymbol{s})^\top \boldsymbol{W}\boldsymbol{A} \quad \text{for symmetric } \boldsymbol{W}$$

Automatic differentiation

Sometimes the derivative of a complex function (a function composed of many small functions) is even more complex than the function itself. Suppose f is such a function composed of n simple functions, as $f = f_n \circ f_{n-1} \dots f_2 \circ f_1$ and f' is composed of $m > n$ simple functions. Then instead of computing the derivative directly, we can keep track of the values of k succesive functions applied on the current input as $x_k = f_k \circ f_{k-1} \circ f_{k-2} \dots f_2 \circ f_1(x)$. Then using $\frac{dx_k}{dx} = f_k'(x_{k-1})\frac{dx_{k-1}}{dx}$, we calculate the derivatives of all x_k and update them using Euler's method (a numerical method for simulating differential equations), then in the last iteration of every update, we arrive at $f'(x) = \frac{dx_n}{dx}$. Here, we've done only n simple computations rather than m computations done when using the explicit form of f' . Note that it's important that you update every x_i only after you have used it to calculate $f'(x_{k-1})$. Also in practice, the equation you would use is not $\frac{dx_k}{dx} = f_k'(x_{k-1})\frac{dx_{k-1}}{dx}$, but just $\Delta x_k = f'(x_{k-1})\Delta x_{k-1}$.

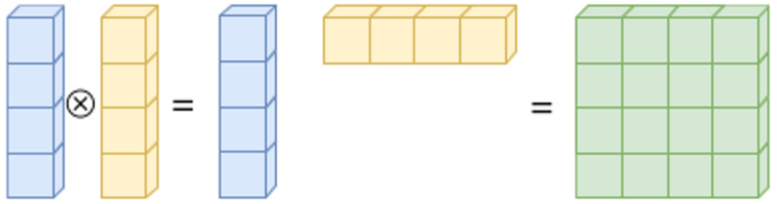
This can also be generalised to functions composed of vector valued simple functions with vector valued inputs.

Second order derivative wrt a vector

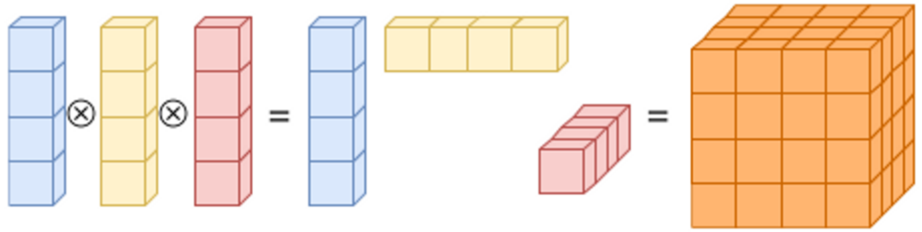
For a scalar valued function $f(\mathbf{x})$ with inputs $\mathbf{x} \in \mathbb{R}^n$, the first order derivative is $\frac{df}{d\mathbf{x}} = [\frac{\partial f}{\partial x_i}]_i^T$ where the square bracket means "stack one below other for different i" , i.e. create a column vector with such and such entries. Notice that the structure in which the vectors are represented doesn't matter all that much, the containment does. (if you have ever used the numpy library in python, you know what I'm talking about) So now, the 2nd derivative is $\frac{d}{d\mathbf{x}}([\frac{\partial f}{\partial x_i}]_i) = [\frac{\partial^2 f}{\partial x_i \partial x_j}]_{ij} = \mathbf{H}$, also known as the Hessian.

You can also think of this as $\nabla^T \nabla f$ where $\nabla = [\frac{\partial}{\partial x_1} \; \frac{\partial}{\partial x_2} \; \cdots \; \frac{\partial}{\partial x_n}]$.

To go to the third derivative, we need a generalised notion of matrices that doesn't depend on the orientation and other trivialities. thus we use tensors and the outer product, which behaves like matrix multiplication :



(a) Given a vector $\boldsymbol{\delta} \in \mathbb{R}^4$, we obtain the outer product $\boldsymbol{\delta}^2 := \boldsymbol{\delta} \otimes \boldsymbol{\delta} = \boldsymbol{\delta}\boldsymbol{\delta}^\top \in \mathbb{R}^{4\times 4}$ as a matrix.



(b) An outer product $\boldsymbol{\delta}^3 := \boldsymbol{\delta} \otimes \boldsymbol{\delta} \otimes \boldsymbol{\delta} \in \mathbb{R}^{4\times 4\times 4}$ results in a third-order tensor ("three-dimensional matrix"), i.e., an array with three indexes.

Now, we define $\nabla^k_{\mathbf{x}}$ to be the k-th total derivative wrt $\mathbf{x}$, which is given by taking k outer products of the nabla operator. Basically, $\nabla^k_{\mathbf{x}} = \nabla \otimes \nabla \otimes \nabla \otimes \ldots \nabla$ (k times).

Multivariate Taylor series

For an analytical function $f(\mathbf{x})$, if $\mathbf{x} = \mathbf{x_0} + \delta$, then we have the taylor series :

$$f(\mathbf{x}) = \sum_{k=0}^{\infty} \frac{\nabla^k f(\mathbf{x_0})}{k!} \cdot \delta^{\mathbf{k}}$$

Here $\delta^{\mathbf{k}} = \delta \otimes \delta \otimes \delta \otimes \ldots \delta$ is the outer product of δ taken k times. we are taking the inner product of $\nabla^k f(\mathbf{x_0})$ and δ^k as

$$\nabla^k f(\mathbf{x_0}) \cdot \delta^k = \sum_{i_1, i_2 \ldots i_n} (\nabla^k f(\mathbf{x_0}))[i_1, i_2 \ldots i_n]\ \delta^k[i_1, i_2, i_3 \ldots i_n] =$$

$$\sum_{i_1, i_2 \ldots i_n} (\nabla^k f(\mathbf{x_0}))[i_1, i_2 \ldots i_n]\ \delta_{i_1}\, \delta_{i_2}\, \delta_{i_3} \ldots \delta_{i_n} =$$

$$\sum_{i_1, i_2 \ldots i_n} (\frac{\partial}{\partial x_{i_1}} \frac{\partial}{\partial x_{i_2}} \frac{\partial}{\partial x_{i_3}} \ldots \frac{\partial}{\partial x_{i_n}} f)_{(\mathbf{x_0})}\ \delta_{i_1}\, \delta_{i_2}\, \delta_{i_3} \ldots \delta_{i_n}$$