Mathematics for Machine Learning

Prof Willie Brink

Applied Mathematics, Stellenbosch University

Lecture 4: Vector Calculus

Contents of the module

5.1 Differentiation of univariate functions

For now we think of a function as a mapping $f : \mathbb{R}^D \to \mathbb{R}$, e.g. $y = f(\boldsymbol{x})$.

The derivative of a univariate $(D = 1)$ function f at x: $\dfrac{df}{dx} = \lim_{h \to 0} \dfrac{f(x+h) - f(x)}{h}$

f' is df/dx, $f^{(2)}$ is the derivative of f', ..., $f^{(k)}$ is the kth derivative of f

The Taylor polynomial of degree n of $f : \mathbb{R} \to \mathbb{R}$ is $T_n(x) = \sum_{k=0}^{n} \dfrac{f^{(k)}(x_0)}{k!}(x - x_0)^k$

It is an approximation of f around x_0.

For an infinitely differentiable f, the Taylor series at x_0 is obtained when $n \to \infty$.

Product: $(f(x)g(x))' = f'(x)g(x) + f(x)g'(x)$.

Quotient: $\left(\frac{f(x)}{g(x)}\right)' = \frac{f'(x)g(x) - f(x)g'(x)}{(g(x))^2}$.

Sum: $(f(x) + g(x))' = f'(x) + g'(x)$.

Chain: $(g(f(x)))' = (g \circ f)'(x) = g'(f(x))f'(x)$.

5.2 Partial differentiation and gradients

The partial derivatives of $f : \mathbb{R}^n \to \mathbb{R}$ of n variables $\boldsymbol{x} = (x_1, \ldots, x_n)$ are

$$\frac{\partial f}{\partial x_1} = \lim_{h \to 0} \frac{f(x_1 + h, x_2, \ldots, x_n) - f(\boldsymbol{x})}{h}, \quad \ldots, \quad \frac{\partial f}{\partial x_n} = \lim_{h \to 0} \frac{f(x_1, \ldots, x_{n-1}, x_n + h) - f(\boldsymbol{x})}{h}$$

The gradient (or Jacobian) of f is the row vector

$$\nabla_{\boldsymbol{x}} f = \operatorname{grad} f = \frac{df}{d\boldsymbol{x}} = \begin{bmatrix} \frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2} & \cdots & \frac{\partial f}{\partial x_n} \end{bmatrix} \in \mathbb{R}^{1 \times n}$$

Let $f : \mathbb{R}^2 \to \mathbb{R}$ be a function of x_1 and x_2, and suppose $x_1(t)$ and $x_2(t)$ are functions of t. Then

$$\begin{aligned} \frac{df}{dt} &= \frac{df}{d\boldsymbol{x}} \frac{d\boldsymbol{x}}{dt} \\ &= \begin{bmatrix} \frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2} \end{bmatrix} \begin{bmatrix} \frac{\partial x_1}{\partial t} \\ \frac{\partial x_2}{\partial t} \end{bmatrix} \\ &= \frac{\partial f}{\partial x_1} \frac{\partial x_1}{\partial t} + \frac{\partial f}{\partial x_2} \frac{\partial x_2}{\partial t} \end{aligned}$$

Product rule: $$\frac{\partial}{\partial \boldsymbol{x}}(f(\boldsymbol{x})g(\boldsymbol{x})) = \frac{\partial f}{\partial \boldsymbol{x}} g(\boldsymbol{x}) + \frac{\partial g}{\partial \boldsymbol{x}} f(\boldsymbol{x})$$

Chain rule: $$\frac{\partial}{\partial \boldsymbol{x}}(g(f(\boldsymbol{x}))) = \frac{\partial}{\partial \boldsymbol{x}}(g \circ f)(\boldsymbol{x}) = \frac{\partial g}{\partial f} \frac{\partial f}{\partial \boldsymbol{x}}$$

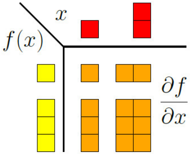
5.3 Gradients of vector-valued functions

Next we generalise the concept of a gradient to vector-valued functions $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^m$.

For such an $\boldsymbol{f}$, and vector $\boldsymbol{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^n$, we have $\boldsymbol{f}(\boldsymbol{x}) = \begin{bmatrix} f_1(\boldsymbol{x}) \\ \vdots \\ f_m(\boldsymbol{x}) \end{bmatrix} \in \mathbb{R}^m$.

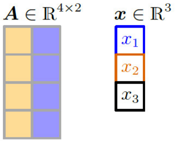
The Jacobian of $\boldsymbol{f}$ is the gradient of $\boldsymbol{f}$ with respect to $\boldsymbol{x}$:

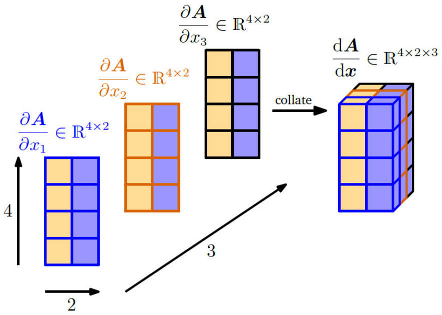
$$\boldsymbol{J} = \nabla_{\boldsymbol{x}} \boldsymbol{f} = \frac{d\boldsymbol{f}}{d\boldsymbol{x}} = \begin{bmatrix} \frac{\partial \boldsymbol{f}}{\partial x_1} & \cdots & \frac{\partial \boldsymbol{f}}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$



5.4 Gradients of matrices

Computing the gradient of an $m \times n$ matrix $\boldsymbol{A}$ with respect to a $p \times q$ matrix $\boldsymbol{B}$ results in a $(m \times n) \times (p \times q)$ Jacobian tensor with elements $J_{ijk\ell} = \partial A_{ij} / \partial B_{k\ell}$.





5.5 Useful identities for computing gradients

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

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

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

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

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

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

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

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

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

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

5.6 Backpropagation and automatic differentiation

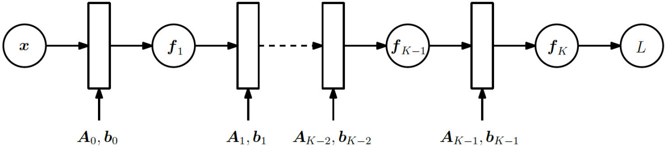
When training deep neural networks we often use gradient descent to find parameters that minimise a loss function. This requires the computation of gradients, for which backpropagation is particularly efficient.

Consider a network with K layers, mapping input $\boldsymbol{x} = \boldsymbol{f}_0$ to output $\boldsymbol{y} = \boldsymbol{f}_K$ as follows:

$$\boldsymbol{f}_i = \sigma_i(\boldsymbol{A}_{i-1}\boldsymbol{f}_{i-1} + \boldsymbol{b}_{i-1}), \quad i = 1, \ldots, K$$

We want parameters $\boldsymbol{\theta} = \{\boldsymbol{A}_0, \boldsymbol{b}_0, \ldots, \boldsymbol{A}_{K-1}, \boldsymbol{b}_{K-1}\}$ that minimise the squared loss

$$L(\boldsymbol{\theta}) = \|\boldsymbol{y} - \boldsymbol{f}_K(\boldsymbol{\theta}, \boldsymbol{x})\|^2$$



To find the gradients w.r.t. parameters $\boldsymbol{\theta}$, we need the partial derivatives of L w.r.t. the parameters $\boldsymbol{\theta}_j = \{\boldsymbol{A}_j, \boldsymbol{b}_j\}$ of each layer $j = 0, \ldots, K-1$.

Using the chain rule,

$$\frac{\partial L}{\partial \boldsymbol{\theta}_{K-1}} = \frac{\partial L}{\partial \boldsymbol{f}_K} \frac{\partial \boldsymbol{f}_K}{\partial \boldsymbol{\theta}_{K-1}}$$

$$\frac{\partial L}{\partial \boldsymbol{\theta}_{K-2}} = \frac{\partial L}{\partial \boldsymbol{f}_K} \boxed{\frac{\partial \boldsymbol{f}_K}{\partial \boldsymbol{f}_{K-1}} \frac{\partial \boldsymbol{f}_{K-1}}{\partial \boldsymbol{\theta}_{K-2}}}$$

$$\frac{\partial L}{\partial \boldsymbol{\theta}_{K-3}} = \frac{\partial L}{\partial \boldsymbol{f}_K} \frac{\partial \boldsymbol{f}_K}{\partial \boldsymbol{f}_{K-1}} \boxed{\frac{\partial \boldsymbol{f}_{K-1}}{\partial \boldsymbol{f}_{K-2}} \frac{\partial \boldsymbol{f}_{K-2}}{\partial \boldsymbol{\theta}_{K-3}}} \qquad \text{etc.}$$

Most of the computation for $\partial L / \partial \boldsymbol{\theta}_{i+1}$ can be reused when computing $\partial L / \partial \boldsymbol{\theta}_i$.

Automatic differentiation: decompose a complicated function (or programme) into a computational graph of primitive operations.

Backprop through the graph, for efficient and accurate calculation of the gradient!

5.7 Higher-order derivatives

Consider a function $f : \mathbb{R}^2 \to \mathbb{R}$ of two variables x and y.

Second-order partial derivatives of f: $\dfrac{\partial^2 f}{\partial x^2}, \ \dfrac{\partial^2 f}{\partial y^2}, \ \dfrac{\partial^2 f}{\partial x \partial y}, \ \dfrac{\partial^2 f}{\partial y \partial x}$

If f is twice continuously differentiable, then $\dfrac{\partial^2 f}{\partial x \partial y} = \dfrac{\partial^2 f}{\partial y \partial x}$

The Hessian collects all second-order partial derivatives: $\boldsymbol{H} = \begin{bmatrix} \dfrac{\partial^2 f}{\partial x^2} & \dfrac{\partial^2 f}{\partial x \partial y} \\ \dfrac{\partial^2 f}{\partial x \partial y} & \dfrac{\partial^2 f}{\partial y^2} \end{bmatrix}$

If $f : \mathbb{R}^n \to \mathbb{R}^m$, the Hessian is an $(m \times n \times n)$ tensor.

5.8 Linearisation and multivariate Taylor series

The gradient ∇f is often used for a locally linear approximation of f around $\boldsymbol{x}_0$:

$$f(\boldsymbol{x}) \approx f(\boldsymbol{x}_0) + (\nabla_{\boldsymbol{x}} f)(\boldsymbol{x}_0)(\boldsymbol{x} - \boldsymbol{x}_0) \quad \text{where } (\nabla_{\boldsymbol{x}} f)(\boldsymbol{x}_0) \text{ is the gradient of } f \text{ at } \boldsymbol{x}_0$$

This is the first-order truncation of the multivariate Taylor series expansion of f at $\boldsymbol{x}_0$:

$$f(\boldsymbol{x}) = \sum_{k=0}^{\infty} \frac{D_{\boldsymbol{x}}^k f(\boldsymbol{x}_0)}{k!} \boldsymbol{\delta}^k$$

where $D_{\boldsymbol{x}}^k f(\boldsymbol{x}_0)$ is the k-th (total) derivative of f with respect to $\boldsymbol{x}$, evaluated at $\boldsymbol{x}_0$, and $\boldsymbol{\delta}^k$ is a k-fold outer product of the vector $\boldsymbol{\delta} = (\boldsymbol{x} - \boldsymbol{x}_0)$.