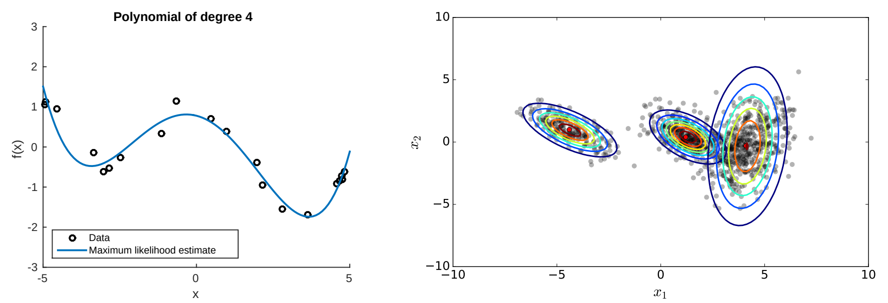
5

Vector Calculus

Many algorithms in machine learning are inherently based on optimizing an objective function with respect to a set of desired model parameters that control how well a model explains the data: Finding good parameters can be phrased as an optimization problem. Examples include linear regression (see Chapter 9), where look at curve-fitting problems, and we optimize linear weight parameters to maximize the likelihood; neural-network auto-encoders for dimensionality reduction and data compression, where the parameters are the weights and biases of each layer, and where we minimize a reconstruction error by repeated application of the chain-rule; Gaussian mixture models (see Chapter 12) for modeling data distributions, where we optimize the location and shape parameters of each mixture component to maximize the likelihood of the model. Figure 5.1 illustrates some of these problems, which we typically solve by using optimization algorithms that exploit gradient information (first-order methods). Figure 5.2 gives an overview of how concepts in this chapter are related and how they are connected to other chapters of the book.

In this chapter, we will discuss how to compute gradients of functions, which is often essential to facilitate learning in machine learning models. Therefore, vector calculus is one of the fundamental mathematical tools we need in machine learning.



(a) Regression problem: Find parameters, such that the curve explains the observations (circles) well.

(b) Density estimation with a Gaussian mixture model: Find means and covariances, such that the data (dots) can be explained well.

Figure 5.1 Vector calculus plays a central role in (a) regression (curve fitting) and (b) density estimation, i.e., modeling data distributions.

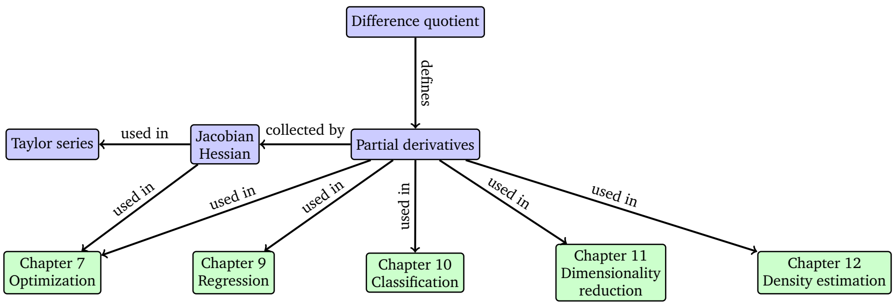


Figure 5.2 A mind map of the concepts introduced in this chapter, along with when they are used in other parts of the book.

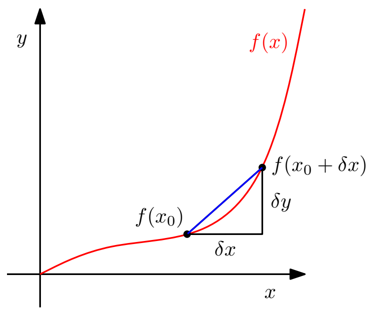


Figure 5.3 The average incline of a function f between x_0 and $x_0 + \delta x$ is the incline of the secant (blue) through $f(x_0)$ and $f(x_0 + \delta x)$ and given by $\delta y / \delta x$.

5.1 Differentiation of Univeriate Functions

In the following, we briefly revisit differentiation of a univariate function, which we may already know from school. We start with the difference quotient of a univariate function $y = f(x)$, $x, y \in \mathbb{R}$, which we will subsequently use to define derivatives.

difference quotient

Definition 5.1 (Difference Quotient)**.** The *difference quotient*

$$\frac{\delta y}{\delta x} := \frac{f(x + \delta x) - f(x)}{\delta x} \tag{5.1}$$

computes the slope of the secant line through two points on the graph of f. In Figure 5.3 these are the points with x-coordinates x_0 and $x_0 + \delta x$.

The difference quotient can also be considered the average slope of f between x and $x + \delta x$ if we assume a f to be a linear function. In the limit for $\delta x \to 0$, we obtain the tangent of f at x, if f is differentiable. The tangent is then the derivative of f at x.

derivative

Definition 5.2 (Derivative)**.** More formally, for $h > 0$ the *derivative* of f at x is defined as the limit

$$\frac{df}{dx} := \lim_{h \to 0} \frac{f(x + h) - f(x)}{h}, \tag{5.2}$$

and the secant in Figure 5.3 becomes a tangent.

Example (Derivative of a Polynomial)

We want to compute the derivative of $f(x) = x^n, n \in \mathbb{N}$. We may already know that the answer will be nx^{n-1}, but we want to derive this result using the definition of the derivative as the limit of the difference quotient.

Using the definition of the derivative in (5.2) we obtain

$$\frac{df}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \tag{5.3}$$
$$= \lim_{h \to 0} \frac{(x+h)^n - x^n}{h} \tag{5.4}$$
$$= \lim_{h \to 0} \frac{\sum_{i=0}^{n} \binom{n}{i} x^{n-i} h^i - x^n}{h} . \tag{5.5}$$

We see that $x^n = \binom{n}{0} x^{n-0} h^0$. By starting the sum at 1 the x^n-term cancels, and we obtain

$$\frac{df}{dx} = \lim_{h \to 0} \frac{\sum_{i=1}^{n} \binom{n}{i} x^{n-i} h^i}{h} \tag{5.6}$$
$$= \lim_{h \to 0} \sum_{i=1}^{n} \binom{n}{i} x^{n-i} h^{i-1} \tag{5.7}$$
$$= \lim_{h \to 0} \binom{n}{1} x^{n-1} + \underbrace{\sum_{i=2}^{n} \binom{n}{i} x^{n-i} h^{i-1}}_{\to 0 \text{ as } h \to 0} \tag{5.8}$$
$$= \frac{n!}{1!(n-1)!} x^{n-1} = nx^{n-1} . \tag{5.9}$$

5.1.1 Taylor Series

The Taylor series is a representation of a function f as an (infinite) sum of terms. These terms are determined using derivatives of f, evaluated at a point x_0.

Definition 5.3 (Taylor Series). For a function $f : \mathbb{R} \to \mathbb{R}$, the *Taylor series* of a smooth function $f \in \mathcal{C}^\infty$ at x_0 is defined as

Taylor series

$f \in \mathcal{C}^\infty$ means that f is infinitely often continuously differentiable.

$$f(x) = \sum_{k=0}^{\infty} \frac{f^{(k)}(x_0)}{k!} (x - x_0)^k , \tag{5.10}$$

where $f^{(k)}(x_0)$ is the kth derivative of f at x_0 and $\frac{f^{(k)}(x_0)}{k!}$ are the coefficients of the polynomial. For $x_0 = 0$, we obtain the *Maclaurin series* as a special instance of the Taylor series.

Maclaurin series

Definition 5.4 (Taylor Polynomial). The *Taylor polynomial* of degree n of

Taylor polynomial

f at x_0 contains the first $n+1$ terms of the series in (5.10) and is defined as

$$T_n := \sum_{k=0}^{n} \frac{f^{(k)}(x_0)}{k!}(x - x_0)^k\,. \tag{5.11}$$

Remark. In general, a Taylor polynomial of degree n is an approximation of a function, which does not need to be a polynomial. The Taylor polynomial is similar to f in a neighborhood around x_0. However, a Taylor polynomial of degree n is an exact representation of a polynomial f of degree $k \leqslant n$ since all derivatives $f^{(i)}$, $i > k$ vanish. $\diamondsuit$

Example (Taylor Polynomial)
We consider the polynomial

$$f(x) = x^4 \tag{5.12}$$

and seek the Taylor polynomial T_6, evaluated at $x_0 = 1$. We start by computing the coefficients $f^{(k)}(1)$ for $k = 0, \ldots, 6$:

$$f(1) = 1 \tag{5.13}$$
$$f'(1) = 4 \tag{5.14}$$
$$f''(1) = 12 \tag{5.15}$$
$$f^{(3)}(1) = 24 \tag{5.16}$$
$$f^{(4)}(1) = 24 \tag{5.17}$$
$$f^{(5)}(1) = 0 \tag{5.18}$$
$$f^{(6)}(1) = 0 \tag{5.19}$$

Therefore, the desired Taylor polynomial is

$$\begin{aligned} T_6(x) &= \sum_{k=0}^{6} \frac{f^{(k)}(x_0)}{k!}(x - x_0)^k && (5.20)\\ &= 1 + 4(x-1) + 6(x-1)^2 + 4(x-1)^3 + (x-1)^4 + 0\,. && (5.21) \end{aligned}$$

Multiplying out and re-arranging yields

$$\begin{aligned} T_6(x) &= (1 - 4 + 6 - 4 + 1) + x(4 - 12 + 12 - 4) \\ &\quad + x^2(6 - 12 + 6) + x^3(4 - 4) + x^4 && (5.22)\\ &= x^4 = f(x)\,, && (5.23) \end{aligned}$$

i.e., we obtain an exact representation of the original function.

Example (Taylor Series)

Consider the smooth function

$$f(x) = \sin(x) + \cos(x) \in \mathcal{C}^{\infty} . \tag{5.24}$$

We seek a Taylor series expansion of f at $x_0 = 0$, which is the Maclaurin series expansion of f. We obtain the following derivatives:

$$\begin{aligned}
f(0) &= \sin(0) + \cos(0) = 1 & (5.25)\\
f'(0) &= \cos(0) - \sin(0) = 1 & (5.26)\\
f''(0) &= -\sin(0) - \cos(0) = -1 & (5.27)\\
f^{(3)}(0) &= -\cos(0) + \sin(0) = -1 & (5.28)\\
f^{(4)}(0) &= \sin(0) + \cos(0) = f(0) = 1 & (5.29)\\
&\vdots
\end{aligned}$$

We can see a pattern here: The coefficients in our Taylor series are only ± 1 (since $\sin(0) = 0$), each of which occurs twice before switching to the other one. Furthermore, $f^{(k+4)}(0) = f^{(k)}(0)$.

Therefore, the full Taylor series expansion of f at $x_0 = 0$ is given by

$$\begin{aligned}
f(x) &= \sum_{k=0}^{\infty} \frac{f^{(k)}(x_0)}{k!}(x - x_0)^k & (5.30)\\
&= 1 + x - \frac{1}{2!}x^2 - \frac{1}{3!}x^3 + \frac{1}{4!}x^4 + \frac{1}{5!}x^5 - \cdots & (5.31)\\
&= 1 - \frac{1}{2!}x^2 + \frac{1}{4!}x^4 \mp \cdots + x - \frac{1}{3!}x^3 + \frac{1}{5!}x^5 \mp \cdots & (5.32)\\
&= \sum_{k=0}^{\infty}(-1)^k \frac{1}{(2k)!}x^{2k} + \sum_{k=0}^{\infty}(-1)^k \frac{1}{(2k+1)!}x^{2k+1} & (5.33)\\
&= \cos(x) + \sin(x) , & (5.34)
\end{aligned}$$

where we used the *power series representations*

power series representations

$$\begin{aligned}
\cos(x) &= \sum_{k=0}^{\infty}(-1)^k \frac{1}{(2k)!}x^{2k} , & (5.35)\\
\sin(x) &= \sum_{k=0}^{\infty}(-1)^k \frac{1}{(2k+1)!}x^{2k+1} . & (5.36)
\end{aligned}$$

Figure 5.4 shows the corresponding first Taylor polynomials T_n for $n = 0, 1, 5, 10$.

5.1.2 Differentiation Rules

In the following, we briefly state basic differentiation rules, where we denote the derivative of f by f'.

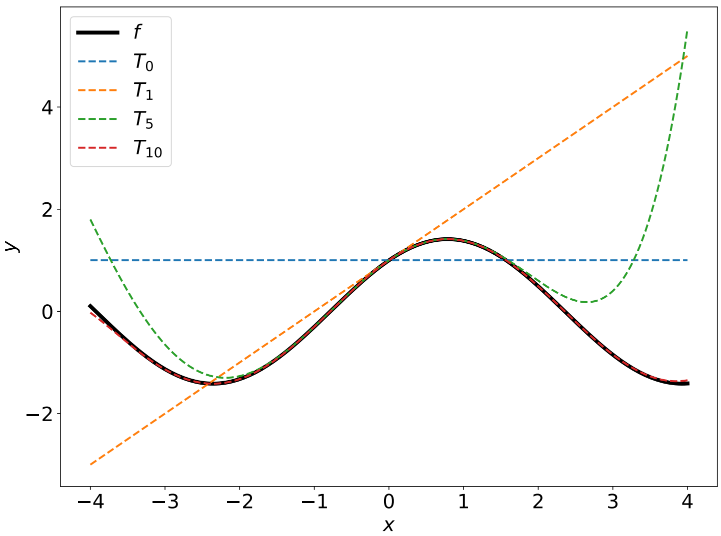


Figure 5.4 Taylor polynomials. The original function $f(x) = \sin(x) + \cos(x)$ (black, solid) is approximated by Taylor polynomials (dashed) around $x_0 = 0$. Higher-order Taylor polynomials approximate the function f better and more globally. T_{10} is already similar to f in $[-4, 4]$.

$$\text{Product Rule:} \quad (f(x)g(x))' = f'(x)g(x) + f(x)g'(x) \tag{5.37}$$

$$\text{Quotient Rule:} \quad \left(\frac{f(x)}{g(x)}\right)' = \frac{f'(x)g(x) - f(x)g'(x)}{(g(x))^2} \tag{5.38}$$

$$\text{Sum Rule:} \quad (f(x) + g(x))' = f'(x) + g'(x) \tag{5.39}$$

$$\text{Chain Rule:} \quad \big(g(f(x))\big)' = (g \circ f)'(x) = g'(f)f'(x) \tag{5.40}$$

Here, $g \circ f$ is a function composition $x \mapsto f(x) \mapsto g(f(x))$.

Example (Chain rule)
Let us compute the derivative of the function $h(x) = (2x+1)^4$ using the chain rule. With

$$h(x) = (2x+1)^4 = g(f(x)), \tag{5.41}$$
$$f(x) = 2x + 1, \tag{5.42}$$
$$g(f) = f^4 \tag{5.43}$$

we obtain the derivatives of f and g as

$$f'(x) = 2, \tag{5.44}$$
$$g'(f) = 4f^3, \tag{5.45}$$

such that the derivative of h is given as

$$h'(x) = g'(f)f'(x) = (4f^3) \cdot 2 \overset{(5.42)}{=} 4(2x+1) \cdot 2 = 8(2x+1), \tag{5.46}$$

where we used the chain rule, see (5.40), and substituted the definition of f in (5.42) in $g'(f)$.

5.2 Partial Differentiation and Gradients

Differentiation as discussed in Section 5.1 applies to functions f of a scalar variable $x \in \mathbb{R}$. In the following, we consider the general case where the function f depends on one or more variables $\boldsymbol{x} \in \mathbb{R}^n$, e.g., $f(\boldsymbol{x}) = f(x_1, x_2)$. The generalization of the derivative to functions of several variables is the *gradient*.

We find the gradient of the function f with respect to $\boldsymbol{x}$ by *varying one variable at a time* and keeping the others constant. The gradient is then the collection of these *partial derivatives*.

Definition 5.5 (Partial Derivative)**.** For a function $f : \mathbb{R}^n \to \mathbb{R}$, $\boldsymbol{x} \mapsto f(\boldsymbol{x})$, $\boldsymbol{x} \in \mathbb{R}^n$ of n variables $x_1, \ldots, x_n$ we define the *partial derivatives* as

partial derivatives

$$\begin{aligned} \frac{\partial f}{\partial x_1} &= \lim_{h \to 0} \frac{f(x_1 + h, x_2, \ldots, x_n) - f(\boldsymbol{x})}{h} \\ &\vdots \\ \frac{\partial f}{\partial x_n} &= \lim_{h \to 0} \frac{f(x_1, \ldots, x_{n-1}, x_n + h) - f(\boldsymbol{x})}{h} \end{aligned} \tag{5.47}$$

and collect them in the row vector

$$\nabla_{\boldsymbol{x}} f = \mathrm{grad} f = \frac{df}{d\boldsymbol{x}} = \begin{bmatrix} \frac{\partial f(\boldsymbol{x})}{\partial x_1} & \frac{\partial f(\boldsymbol{x})}{\partial x_2} & \cdots & \frac{\partial f(\boldsymbol{x})}{\partial x_n} \end{bmatrix} \in \mathbb{R}^{1 \times n}, \tag{5.48}$$

where n is the number of variables and 1 is the dimension of the image/range of f. Here, we used the compact vector notation $\boldsymbol{x} = [x_1, \ldots, x_n]^\top$. The row vector in (5.48) is called the *gradient* of f or the *Jacobian* and is the generalization of the derivative from Section 5.1.

gradient

Jacobian

Example (Partial Derivatives using the Chain Rule)

For $f(x, y) = (x + 2y^3)^2$, we obtain the partial derivatives

We can use results from scalar differentiation: Each partial derivative is a derivative with respect to a scalar.

$$\frac{\partial f(x, y)}{\partial x} = 2(x + 2y^3)\frac{\partial}{\partial x}(x + 2y^3) = 2(x + 2y^3), \tag{5.49}$$

$$\frac{\partial f(x, y)}{\partial y} = 2(x + 2y^3)\frac{\partial}{\partial y}(x + 2y^3) = 12(x + 2y^3)y^2. \tag{5.50}$$

where we used the chain rule (5.40) to compute the partial derivatives.

Remark (Gradient as a Row Vector). It is not uncommon in the literature to define the gradient vector as a column vector, following the convention that vectors are generally column vectors. The reason why we define the gradient vector as a row vector is twofold: First, we can consistently generalize the gradient to a setting where $f : \mathbb{R}^n \to \mathbb{R}^m$ no longer maps onto the real line (then the gradient becomes a matrix). Second, we can immediately apply the multi-variate chain-rule without paying attention to the dimension of the gradient. We will discuss both points later. $\diamondsuit$

Example (Gradient)
For $f(x_1, x_2) = x_1^2 x_2 + x_1 x_2^3 \in \mathbb{R}$, the partial derivatives (i.e., the derivatives of f with respect to x_1 and x_2) are

$$\frac{\partial f(x_1, x_2)}{\partial x_1} = 2x_1 x_2 + x_2^3 \tag{5.51}$$

$$\frac{\partial f(x_1, x_2)}{\partial x_2} = x_1^2 + 3x_1 x_2^2 \tag{5.52}$$

and the gradient is then

$$\frac{df}{d\boldsymbol{x}} = \begin{bmatrix} \frac{\partial f(x_1,x_2)}{\partial x_1} & \frac{\partial f(x_1,x_2)}{\partial x_2} \end{bmatrix} = \begin{bmatrix} 2x_1x_2 + x_2^3 & x_1^2 + 3x_1x_2^2 \end{bmatrix} \in \mathbb{R}^{1\times 2}\,. \tag{5.53}$$

5.2.1 Basic Rules of Partial Differentiation

Product rule: $(fg)' = f'g + fg'$
Sum rule: $(f+g)' = f' + g'$
Chain rule: $(g(f))' = g'(f)f'$

In the multivariate case, where $\boldsymbol{x} \in \mathbb{R}^n$, the basic differentiation rules that we know from school (e.g., sum rule, product rule, chain rule; see also Section 5.1.2) still apply. However, when we compute derivatives with respect to vectors $\boldsymbol{x} \in \$^n$ we need to pay attention: Our gradients now involve vectors and matrices, and matrix multiplication is no longer commutative (see Section 2.2.1), i.e., the order matters.

Here are the general product rule, sum rule and chain rule:

$$\text{Product Rule:} \quad \frac{\partial}{\partial \boldsymbol{x}}\big(f(\boldsymbol{x})g(\boldsymbol{x})\big) = \frac{\partial f}{\partial \boldsymbol{x}} g(\boldsymbol{x}) + f(\boldsymbol{x})\frac{\partial g}{\partial \boldsymbol{x}} \tag{5.54}$$

$$\text{Sum Rule:} \quad \frac{\partial}{\partial \boldsymbol{x}}\big(f(\boldsymbol{x}) + g(\boldsymbol{x})\big) = \frac{\partial f}{\partial \boldsymbol{x}} + \frac{\partial g}{\partial \boldsymbol{x}} \tag{5.55}$$

$$\text{Chain Rule:} \quad \frac{\partial}{\partial \boldsymbol{x}}(g \circ f)(\boldsymbol{x}) = \frac{\partial}{\partial \boldsymbol{x}}\big(g(f(\boldsymbol{x}))\big) = \frac{\partial g}{\partial f}\frac{\partial f}{\partial \boldsymbol{x}} \tag{5.56}$$

This is only an intuition, but not mathematically correct since the partial derivative is not a fraction.

Let us have a closer look at the chain rule. The chain rule (5.56) resembles to some degree the rules for matrix multiplication where we said that neighboring dimensions have to match for matrix multiplication to be defined, see Section 2.2.1. If we go from left to right, the chain rule exhibits similar properties: ∂f shows up in the "denominator" of the first factor and in the "numerator" of the second factor. If we multiply the factors together, multiplication is defined (the dimensions of ∂f match, and ∂f "cancels", such that $\partial g/\partial \boldsymbol{x}$ remains.

5.2.2 Chain Rule

Consider a function $f : \mathbb{R}^2 \to \mathbb{R}$ of two variables x_1, x_2. Furthermore, $x_1(t)$ and $x_2(t)$ are themselves functions of t. To compute the gradient of f with respect to t, we need to apply the chain rule (5.56) for multivariate functions as

$$\frac{df}{dt} = \begin{bmatrix} \frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2} \end{bmatrix} \begin{bmatrix} \frac{\partial x_1(t)}{\partial t} \\ \frac{\partial x_2(t)}{\partial t} \end{bmatrix} = \frac{\partial f}{\partial x_1}\frac{\partial x_1}{\partial t} + \frac{\partial f}{\partial x_2}\frac{\partial x_2}{\partial t} \tag{5.57}$$

where d denotes the gradient and ∂ partial derivatives.

Example

Consider $f(x_1, x_2) = x_1^2 + 2x_2$, where $x_1 = \sin t$ and $x_2 = \cos t$, then

$$\frac{df}{dt} = \frac{\partial f}{\partial x_1}\frac{\partial x_1}{\partial t} + \frac{\partial f}{\partial x_2}\frac{\partial x_2}{\partial t} \tag{5.58}$$

$$= 2\sin t \frac{\partial \sin t}{\partial t} + 2\frac{\partial \cos t}{\partial t} \tag{5.59}$$

$$= 2\sin t \cos t - 2\sin t = 2\sin t(\cos t - 1) \tag{5.60}$$

is the corresponding derivative of f with respect to t.

If $f(x_1, x_2)$ is a function of x_1 and x_2, where $x_1(s, t)$ and $x_2(s, t)$ are themselves functions of two variables s and t, the chain rule yields the partial derivatives

$$\frac{\partial f}{\partial s} = \frac{\partial f}{\partial x_1}\frac{\partial x_1}{\partial s} + \frac{\partial f}{\partial x_2}\frac{\partial x_2}{\partial s}, \tag{5.61}$$

$$\frac{\partial f}{\partial t} = \frac{\partial f}{\partial x_1}\frac{\partial x_1}{\partial t} + \frac{\partial f}{\partial x_2}\frac{\partial x_2}{\partial t}, \tag{5.62}$$

and the gradient is obtained by the matrix multiplication

$$\frac{df}{d(s,t)} = \frac{\partial f}{\partial \boldsymbol{x}}\frac{\partial \boldsymbol{x}}{\partial (s,t)} = \underbrace{\begin{bmatrix} \frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2} \end{bmatrix}}_{=\frac{\partial f}{\partial \boldsymbol{x}}} \underbrace{\begin{bmatrix} \frac{\partial x_1}{\partial s} & \frac{\partial x_1}{\partial t} \\ \frac{\partial x_2}{\partial s} & \frac{\partial x_2}{\partial t} \end{bmatrix}}_{=\frac{\partial \boldsymbol{x}}{\partial (s,t)}}. \tag{5.63}$$

This compact way of writing the chain rule as a matrix multiplication makes only sense if the gradient is defined as a row vector. Otherwise, we will need to start transposing gradients for the matrix dimensions to match. This may still be straightforward as long as the gradient is a vector or a matrix; however, when the gradient becomes a tensor (we will discuss this in the following), the transpose is no longer a triviality.

The chain rule can be written as a matrix multiplication.

Remark (Verifying the Correctness of the Implementation of the Gradient). The definition of the partial derivatives as the limit of the corresponding difference quotient, see (5.47), can be exploited when numerically checking the correctness of gradients in computer programs: When we compute gradients and implement them, we can use finite differences to numerically test our computation and implementation: We choose the value h to be small (e.g., $h = 10^{-4}$) and compare the finite-difference approximation from (5.47) with our (analytic) implementation of the gradient. If the error is small, our gradient implementation is probably correct. "Small" could mean that $\sqrt{\frac{\sum_i (dh_i - df_i)^2}{\sum_i (dh_i + df_i)^2}} < 10^{-3}$, where dh_i is the finite-difference approximation and df_i is the analytic gradient of f with respect to the ith variable x_i. ♢

Gradient checking

5.3 Gradients of Vector-Valued Functions (Vector Fields)

Thus far, we discussed partial derivatives and gradients of functions $f : \mathbb{R}^n \to \mathbb{R}$ mapping to the real numbers. In the following, we will generalize the concept of the gradient to vector-valued functions $f : \mathbb{R}^n \to \mathbb{R}^m$, where $n, m \geqslant 1$.

For a function $f : \mathbb{R}^n \to \mathbb{R}^m$ and a vector $\boldsymbol{x} = [x_1, \dots, x_n]^\top \in \mathbb{R}^n$, the corresponding vector of function values is given as

$$\boldsymbol{f}(\boldsymbol{x}) = \begin{bmatrix} f_1(\boldsymbol{x}) \\ \vdots \\ f_m(\boldsymbol{x}) \end{bmatrix} \in \mathbb{R}^m \,. \tag{5.64}$$

Writing the vector-valued function in this way allows us to view a vector-valued function $f : \mathbb{R}^n \to \mathbb{R}^m$ as a vector of functions $[f_1, \dots, f_m]^\top$, $f_i : \mathbb{R}^n \to \mathbb{R}$ that map onto $\mathbb{R}$. The differentiation rules for every f_i are exactly the ones we discussed in Section 5.2.

Therefore, the *partial derivative of a vector-valued function* $f : \mathbb{R}^n \to \mathbb{R}^m$ with respect to $x_i \in \mathbb{R}$, $i = 1, \dots n$, is given as the vector

partial derivative of a vector-valued function

$$\frac{\partial f}{\partial x_i} = \begin{bmatrix} \frac{\partial f_1}{\partial x_i} \\ \vdots \\ \frac{\partial f_m}{\partial x_i} \end{bmatrix} = \begin{bmatrix} \lim_{h \to 0} \frac{f_1(x_1, \dots, x_{i-1}, x_i + h, x_{i+1}, \dots x_n) - f_1(\boldsymbol{x})}{h} \\ \vdots \\ \lim_{h \to 0} \frac{f_m(x_1, \dots, x_{i-1}, x_i + h, x_{i+1}, \dots x_n) - f_m(\boldsymbol{x})}{h} \end{bmatrix} \in \mathbb{R}^m \,. \tag{5.65}$$

From (5.48), we know that we obtain the gradient of f with respect to a vector as the row vector of the partial derivatives. In (5.65), every partial derivative $\partial f / \partial x_i$ is a column vector. Therefore, we obtain the gradient of $f : \mathbb{R}^n \to \mathbb{R}^m$ with respect to $\boldsymbol{x} \in \mathbb{R}^n$ by collecting these partial derivatives:

$$\frac{d\boldsymbol{f}(\boldsymbol{x})}{d\boldsymbol{x}} = \left[\boxed{\frac{\partial \boldsymbol{f}(\boldsymbol{x})}{\partial x_1}} \cdots \boxed{\frac{\partial \boldsymbol{f}(\boldsymbol{x})}{\partial x_n}} \right] = \left[\boxed{\begin{matrix} \frac{\partial f_1(\boldsymbol{x})}{\partial x_1} \\ \vdots \\ \frac{\partial f_m(\boldsymbol{x})}{\partial x_1} \end{matrix}} \cdots \boxed{\begin{matrix} \frac{\partial f_1(\boldsymbol{x})}{\partial x_n} \\ \vdots \\ \frac{\partial f_m(\boldsymbol{x})}{\partial x_n} \end{matrix}} \right] \in \mathbb{R}^{m \times n} . \tag{5.66}$$

Definition 5.6 (Jacobian)**.** The collection of all first-order partial derivatives of a vector-valued function $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^m$ is called the *Jacobian*. The Jacobian $\boldsymbol{J}$ is an $m \times n$ matrix, which we define and arrange as follows:

Jacobian

The gradient of a function $f : \mathbb{R}^n \to \mathbb{R}^m$ is a matrix of size $m \times n$.

$$\boldsymbol{J} = \nabla_{\boldsymbol{x}} \boldsymbol{f} = \frac{d\boldsymbol{f}(\boldsymbol{x})}{d\boldsymbol{x}} = \begin{bmatrix} \frac{\partial \boldsymbol{f}(\boldsymbol{x})}{\partial x_1} & \cdots & \frac{\partial \boldsymbol{f}(\boldsymbol{x})}{\partial x_n} \end{bmatrix} \tag{5.67}$$

$$= \begin{bmatrix} \frac{\partial f_1(\boldsymbol{x})}{\partial x_1} & \cdots & \frac{\partial f_1(\boldsymbol{x})}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial f_m(\boldsymbol{x})}{\partial x_1} & \cdots & \frac{\partial f_m(\boldsymbol{x})}{\partial x_n} \end{bmatrix} , \tag{5.68}$$

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} , \quad J(i, j) = \frac{\partial f_i}{\partial x_j} . \tag{5.69}$$

In particular, a function $f : \mathbb{R}^n \to \mathbb{R}^1$, which maps a vector $\boldsymbol{x} \in \mathbb{R}^n$ onto a scalar (e.g., $f(\boldsymbol{x}) = \sum_{i=1}^n x_i$), possesses a Jacobian that is a row vector (matrix of dimension $1 \times n$), see (5.48).

Example (Gradient of a Vector Field)

We are given

$$\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x} , \qquad \boldsymbol{f}(\boldsymbol{x}) \in \mathbb{R}^M, \quad \boldsymbol{A} \in \mathbb{R}^{M \times N}, \quad \boldsymbol{x} \in \mathbb{R}^N .$$

To compute the gradient $d\boldsymbol{f}/d\boldsymbol{x}$ we first determine the dimension of $d\boldsymbol{f}/d\boldsymbol{x}$: Since $\boldsymbol{f} : \mathbb{R}^N \to \mathbb{R}^M$, it follows that $d\boldsymbol{f}/d\boldsymbol{x} \in \mathbb{R}^{M \times N}$. Second, to compute the gradient compute the partial derivatives of f with respect to every x_j:

$$f_i(\boldsymbol{x}) = \sum_{j=1}^{N} A_{ij} x_j \implies \frac{\partial f_i}{\partial x_j} = A_{ij} \tag{5.70}$$

Finally, we collect the partial derivatives in the Jacobian and obtain the gradient as

$$\frac{d\boldsymbol{f}}{d\boldsymbol{x}} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_N} \\ \vdots & & \vdots \\ \frac{\partial f_M}{\partial x_1} & \cdots & \frac{\partial f_M}{\partial x_N} \end{bmatrix} = \begin{bmatrix} A_{11} & \cdots & A_{1N} \\ \vdots & & \vdots \\ A_{M1} & \cdots & A_{MN} \end{bmatrix} = \boldsymbol{A} \in \mathbb{R}^{M \times N} . \tag{5.71}$$

Example (Chain Rule)

Consider the function $h : \mathbb{R} \to \mathbb{R}$, $h(t) = (f \circ g)(t)$ with

$$f : \mathbb{R}^2 \to \mathbb{R} \tag{5.72}$$
$$g : \mathbb{R} \to \mathbb{R}^2 \tag{5.73}$$
$$f(\boldsymbol{x}) = \exp(x_1 x_2^2)\,, \tag{5.74}$$
$$\boldsymbol{x}\begin{bmatrix}x_1\\x_2\end{bmatrix} = g(t) = \begin{bmatrix}t\cos t\\t\sin t\end{bmatrix} \tag{5.75}$$

and compute the gradient of h with respect to t.

Since $f : \mathbb{R}^2 \to \mathbb{R}$ and $g : \mathbb{R} \to \mathbb{R}^2$ we note that

$$\frac{\partial f}{\partial \boldsymbol{x}} \in \mathbb{R}^{1\times 2}\,, \tag{5.76}$$
$$\frac{\partial g}{\partial t} \in \mathbb{R}^{2\times 1}\,. \tag{5.77}$$

The desired gradient is computed by applying the chain-rule:

$$\frac{dh}{dt} = \frac{\partial f}{\partial \boldsymbol{x}}\frac{\partial \boldsymbol{x}}{\partial t} = \begin{bmatrix}\frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2}\end{bmatrix}\begin{bmatrix}\frac{\partial x_1}{\partial t}\\ \frac{\partial x_2}{\partial t}\end{bmatrix} \tag{5.78}$$
$$= \begin{bmatrix}\exp(x_1x_2^2)x_2^2 & 2\exp(x_1x_2^2)x_1x_2\end{bmatrix}\begin{bmatrix}\cos t - t\sin t\\ \sin t + t\cos t\end{bmatrix} \tag{5.79}$$
$$= \exp(x_1x_2^2)\big(x_2^2(\cos t - t\sin t) + 2x_1x_2(\sin t + t\cos t)\big)\,, \tag{5.80}$$

where $x_1 = t\cos t$ and $x_2 = t\sin t$, see (5.75).

Example (Gradient of a Linear Model)

Let us consider the linear model

$$\boldsymbol{y} = \boldsymbol{\Phi\theta}\,, \tag{5.81}$$

where $\boldsymbol{\theta} \in \mathbb{R}^D$ is a parameter vector, $\boldsymbol{\Phi} \in \mathbb{R}^{N\times D}$ are input features and $\boldsymbol{y} \in \mathbb{R}^N$ are the corresponding observations. We define the following functions:

We will discuss this model in much more detail in Chapter 9 in the context of linear regression.

$$L(\boldsymbol{e}) := \|\boldsymbol{e}\|^2\,, \tag{5.82}$$
$$\boldsymbol{e}(\boldsymbol{\theta}) := \boldsymbol{y} - \boldsymbol{\Phi\theta}\,. \tag{5.83}$$

We seek $\frac{\partial L}{\partial \boldsymbol{\theta}}$, and we will use the chain rule for this purpose.

Before we start any calculation, we determine the dimensionality of the gradient as

$$\frac{\partial L}{\partial \boldsymbol{\theta}} \in \mathbb{R}^{1\times D}\,. \tag{5.84}$$

The chain rule allows us to compute the gradient as

$$\frac{\partial L}{\partial \boldsymbol{\theta}} = \frac{\partial L}{\partial \boldsymbol{e}} \frac{\partial \boldsymbol{e}}{\partial \boldsymbol{\theta}}, \tag{5.85}$$

i.e., every element is given by

```
dLdtheta =
np.einsum('j,ji',dLde,dedtheta)
```

$$\frac{\partial L}{\partial \boldsymbol{\theta}}[1, i] = \sum_{j=1}^{D} \frac{\partial L}{\partial \boldsymbol{e}}[j] \frac{\partial \boldsymbol{e}}{\partial \boldsymbol{\theta}}[j, i]. \tag{5.86}$$

We know that $\|\boldsymbol{e}\|^2 = \boldsymbol{e}^\top \boldsymbol{e}$ (see Section 3.2) and determine

$$\frac{\partial L}{\partial \boldsymbol{e}} = 2\boldsymbol{e}^\top \in \mathbb{R}^{1 \times N}. \tag{5.87}$$

Furthermore, we obtain

$$\frac{\partial \boldsymbol{e}}{\partial \boldsymbol{\theta}} = -\boldsymbol{\Phi} \in \mathbb{R}^{N \times D}, \tag{5.88}$$

such that our desired derivative is

$$\frac{\partial L}{\partial \boldsymbol{\theta}} = -2\boldsymbol{e}^\top \boldsymbol{\Phi} \overset{(5.83)}{=} -\underbrace{2(\boldsymbol{y}^\top - \boldsymbol{\theta}^\top \boldsymbol{\Phi}^\top)}_{1 \times N} \underbrace{\boldsymbol{\Phi}}_{N \times D} \in \mathbb{R}^{1 \times D}. \tag{5.89}$$

Remark. We would have obtained the same result without using the chain rule by immediately looking at the function

$$L_2(\boldsymbol{\theta}) := \|\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{\theta}\|^2 = (\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{\theta})^\top (\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{\theta}). \tag{5.90}$$

This approach is still practical for simple functions like L_2 but becomes impractical if consider deep function compositions. ♢

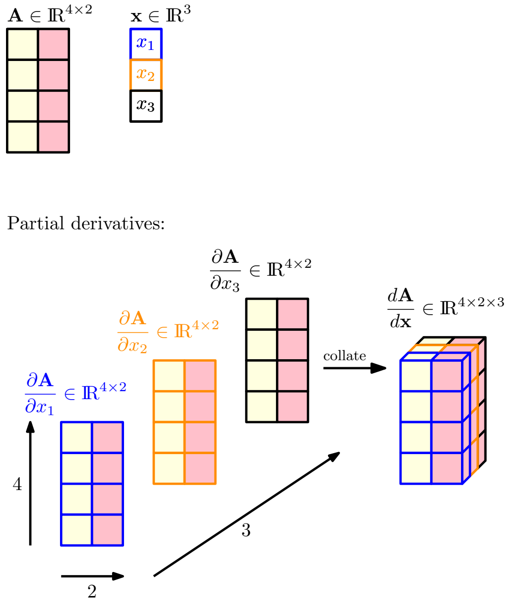
5.4 Gradients of Matrices

We will encounter situations where we need to take gradients of matrices with respect to vectors (or other matrices), which results in a multidimensional tensor. For example, if we compute the gradient of an $m \times n$ matrix with respect to a $p \times q$ matrix, the resulting Jacobian would be $(p \times q) \times (m \times n)$, i.e., a four-dimensional tensor (or array). Since matrices represent linear mappings, we can exploit the fact that there is a vector-space isomorphism (linear, invertible mapping) between the space $\mathbb{R}^{m \times n}$ of $m \times n$ matrices and the space $\mathbb{R}^{mn}$ of mn vectors. Therefore, we can re-shape our matrices into vectors of lengths mn and pq, respectively. The gradient using these mn vectors results in a Jacobian of size $pq \times mn$. Figure 5.5 visualizes both approaches.

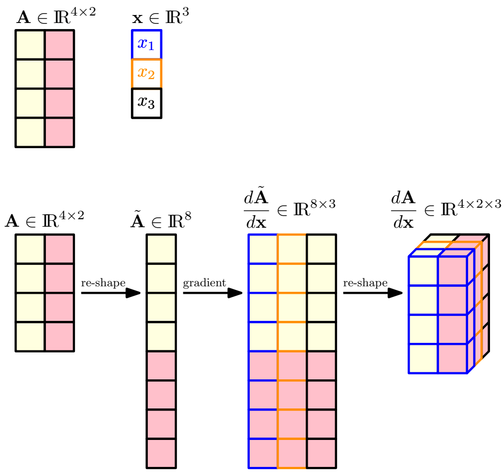
Matrices can be transformed into vectors by stacking the columns of the matrix ("flattening").

In practical applications, it is often desirable to re-shape the matrix

Figure 5.5 Visualization of gradient computation of a matrix with respect to a vector. We are interested in computing the gradient of $\boldsymbol{A} \in \mathbb{R}^{4\times 2}$ with respect to a vector $\boldsymbol{x} \in \mathbb{R}^3$. We know that gradient $\frac{d\boldsymbol{A}}{d\boldsymbol{x}} \in \mathbb{R}^{4\times 2\times 3}$. We follow two equivalent approaches to arrive there: (a) Collating partial derivatives into a Jacobian tensor; (b) Flattening of the matrix into a vector, computing the Jacobian matrix, re-shaping into a Jacobian tensor.



(a) Approach 1: We compute the partial derivative $\frac{\partial \boldsymbol{A}}{\partial x_1}, \frac{\partial \boldsymbol{A}}{\partial x_2}, \frac{\partial \boldsymbol{A}}{\partial x_3}$, each of which is a 4×2 matrix, and collate them in a $4 \times 2 \times 3$ tensor.



(b) Approach 2: We re-shape (flatten) $\boldsymbol{A} \in \mathbb{R}^{4\times 2}$ into a vector $\tilde{\boldsymbol{A}} \in \mathbb{R}^8$. Then, we compute the gradient $\frac{d\tilde{\boldsymbol{A}}}{d\boldsymbol{x}} \in \mathbb{R}^{8\times 3}$. We obtain the gradient tensor by re-shaping this gradient as illustrated above.

into a vector and continue working with this Jacobian matrix: The chain rule (5.56) boils down to simple matrix multiplication, whereas in the case of a Jacobian tensor, we will need to pay more attention to what dimensions we need to sum out.

Example (Gradient of Vectors with Respect to Matrices)
Let us consider the following example, where

$$\boldsymbol{f} = \boldsymbol{A}\boldsymbol{x}\,, \quad \boldsymbol{f} \in \mathbb{R}^M, \boldsymbol{A} \in \mathbb{R}^{M\times N}, \boldsymbol{x} \in \mathbb{R}^N \tag{5.91}$$

and where we seek the gradient $d\boldsymbol{f}/d\boldsymbol{A}$. Let us start again by determining the dimension of the gradient as

$$\frac{d\boldsymbol{f}}{d\boldsymbol{A}} \in \mathbb{R}^{M\times(M\times N)}\,. \tag{5.92}$$

By definition, the gradient is the collection of the partial derivatives:

$$\frac{d\boldsymbol{f}}{d\boldsymbol{A}} = \begin{bmatrix} \frac{\partial f_1}{\partial \boldsymbol{A}} \\ \vdots \\ \frac{\partial f_M}{\partial \boldsymbol{A}} \end{bmatrix}, \quad \frac{\partial f_i}{\partial \boldsymbol{A}} \in \mathbb{R}^{1\times(M\times N)}\,. \tag{5.93}$$

To compute the partial derivatives, it will be helpful to explicitly write out the matrix vector multiplication:

$$f_i = \sum_{j=1}^{N} A_{ij}x_j, \quad i = 1, \ldots, M\,, \tag{5.94}$$

and the partial derivatives are then given as

$$\frac{\partial f_i}{\partial A_{iq}} = x_q\,. \tag{5.95}$$

This allows us to compute the partial derivatives of f_i with respect to a row of $\boldsymbol{A}$, which is given as

$$\frac{\partial f_i}{\partial A_{i,:}} = \boldsymbol{x}^\top \in \mathbb{R}^{1\times1\times N}\,, \tag{5.96}$$

$$\frac{\partial f_i}{\partial A_{k\neq i,:}} = \boldsymbol{0}^\top \in \mathbb{R}^{1\times1\times N} \tag{5.97}$$

where we have to pay attention to the correct dimensionality. Since f_i maps onto $\mathbb{R}$ and each row of $\boldsymbol{A}$ is of size $1\times N$, we obtain a $1\times1\times N$-sized tensor as the partial derivative of f_i with respect to a row of $\boldsymbol{A}$.

We can now stack the partial derivatives to obtain the desired gradient as

$$\frac{\partial f_i}{\partial \boldsymbol{A}} = \begin{bmatrix} \boldsymbol{0}^\top \\ \vdots \\ \boldsymbol{x}^\top \\ \boldsymbol{0}^\top \\ \vdots \\ \boldsymbol{0}^\top \end{bmatrix} \in \mathbb{R}^{1\times(M\times N)}\,. \tag{5.98}$$

Example (Gradient of Matrices with Respect to Matrices)
Consider a matrix $\boldsymbol{L} \in \mathbb{R}^{m\times n}$ and $f : \mathbb{R}^{m\times n} \to \mathbb{R}^{n\times n}$ with

$$\boldsymbol{f}(\boldsymbol{L}) = \boldsymbol{L}^\top \boldsymbol{L} =: \boldsymbol{K} \in \mathbb{R}^{n\times n} . \tag{5.99}$$

where we seek the gradient $d\boldsymbol{K}/d\boldsymbol{L}$. To solve this hard problem, let us first write down what we already know: We know that the gradient has the dimensions

$$\frac{d\boldsymbol{K}}{d\boldsymbol{L}} \in \mathbb{R}^{(n\times n)\times(m\times n)} , \tag{5.100}$$

which is a tensor. If we compute the partial derivative of f with respect to a single entry L_{ij}, $i, j \in \{1, \dots, n\}$, of $\boldsymbol{L}$, we obtain an $n \times n$-matrix

$$\frac{\partial \boldsymbol{K}}{\partial L_{ij}} \in \mathbb{R}^{n\times n} . \tag{5.101}$$

Furthermore, we know that

$$\frac{dK_{pq}}{d\boldsymbol{L}} \in \mathbb{R}^{1\times m\times n} \tag{5.102}$$

for $p, q = 1, \dots, n$, where $K_{pq} = f_{pq}(\boldsymbol{L})$ is the (p, q)-th entry of $\boldsymbol{K} = f(\boldsymbol{L})$.

Denoting the i-th column of $\boldsymbol{L}$ by $\boldsymbol{l}_i$, we see that every entry of $\boldsymbol{K}$ is given by an inner product of two columns of $\boldsymbol{L}$, i.e.,

$$K_{pq} = \boldsymbol{l}_p^\top \boldsymbol{l}_q = \sum_{k=1}^{n} L_{kp} L_{kq} . \tag{5.103}$$

When we now compute the partial derivative $\frac{\partial K_{pq}}{\partial L_{ij}}$, we obtain

$$\frac{\partial K_{pq}}{\partial L_{ij}} = \sum_{k=1}^{n} \frac{\partial}{\partial L_{ij}} L_{kp} L_{kq} = \partial_{pqij} , \tag{5.104}$$

$$\partial_{pqij} = \begin{cases} L_{iq} & \text{if } j = p,\ p \neq q \\ L_{ip} & \text{if } j = q,\ p \neq q \\ 2L_{iq} & \text{if } j = p,\ p = q \\ 0 & \text{otherwise} \end{cases} \tag{5.105}$$

From (5.100), we know that the desired gradient has the dimension $(n \times n) \times (m \times n)$, end every single entry of this tensor is given by ∂_{pqij} in (5.105), where $p, q, j = 1, \dots, n$ and $i = q, \dots, m$.

5.5 Useful Identities for Computing Gradients

In the following, we list some useful gradients that are frequently required in a machine learning context (Petersen and Pedersen, 2012):

$$\frac{\partial}{\partial \boldsymbol{X}} f(\boldsymbol{X})^\top = \left(\frac{\partial f(\boldsymbol{X})}{\partial \boldsymbol{X}}\right)^\top \tag{5.106}$$

$$\frac{\partial}{\partial \boldsymbol{X}} \mathrm{tr}(f(\boldsymbol{X})) = \mathrm{tr}\left(\frac{\partial f(\boldsymbol{X})}{\partial \boldsymbol{X}}\right) \tag{5.107}$$

$$\frac{\partial}{\partial \boldsymbol{X}} \det(f(\boldsymbol{X})) = \det(f(\boldsymbol{X}))\mathrm{tr}\left(f^{-1}(\boldsymbol{X})\frac{\partial f(\boldsymbol{X})}{\partial \boldsymbol{X}}\right) \tag{5.108}$$

$$\frac{\partial}{\partial \boldsymbol{X}} f^{-1}(\boldsymbol{X}) = -f^{-1}(\boldsymbol{X})\frac{\partial f(\boldsymbol{X})}{\partial \boldsymbol{X}} f^{-1}(\boldsymbol{X}) \tag{5.109}$$

$$\frac{\partial \boldsymbol{a}^\top \boldsymbol{X}^{-1}\boldsymbol{b}}{\partial \boldsymbol{X}} = -(\boldsymbol{X}^{-1})^\top \boldsymbol{a}\boldsymbol{b}^\top (\boldsymbol{X}^{-1})^\top \tag{5.110}$$

$$\frac{\partial \boldsymbol{x}^\top \boldsymbol{a}}{\partial \boldsymbol{x}} = \boldsymbol{a}^\top \tag{5.111}$$

$$\frac{\partial \boldsymbol{a}^\top \boldsymbol{x}}{\partial \boldsymbol{x}} = \boldsymbol{a}^\top \tag{5.112}$$

$$\frac{\partial \boldsymbol{a}^\top \boldsymbol{X}\boldsymbol{b}}{\partial \boldsymbol{X}} = \boldsymbol{a}\boldsymbol{b}^\top \tag{5.113}$$

$$\frac{\partial \boldsymbol{x}^\top \boldsymbol{B}\boldsymbol{x}}{\partial \boldsymbol{x}} = \boldsymbol{x}^\top(\boldsymbol{B}+\boldsymbol{B}^\top) \tag{5.114}$$

$$\frac{\partial}{\partial \boldsymbol{s}}(\boldsymbol{x}-\boldsymbol{A}\boldsymbol{s})^\top \boldsymbol{W}(\boldsymbol{x}-\boldsymbol{A}\boldsymbol{s}) = -2(\boldsymbol{x}-\boldsymbol{A}\boldsymbol{s})^\top \boldsymbol{W}\boldsymbol{A} \quad \text{for symmetric } \boldsymbol{W} \tag{5.115}$$

5.6 Backpropagation and Automatic Differentiation

In many machine learning applications, we find good model parameters by performing gradient descent (Chapter 7), which relies on the fact that we can compute the gradient of a learning objective with respect to the parameters of the model. For a given objective function, we can obtain the gradient with respect to the model parameters using calculus and applying the chain rule, see Section 5.2.2. We already had a taster in Section 5.3 when we looked at the gradient of a squared loss with respect to the parameters of a linear regression model.

Consider the function

$$f(x) = \sqrt{x^2 + \exp(x^2)} + \cos\left(x^2 + \exp(x^2)\right). \tag{5.116}$$

By application of the chain rule, and noting that differentiation is linear

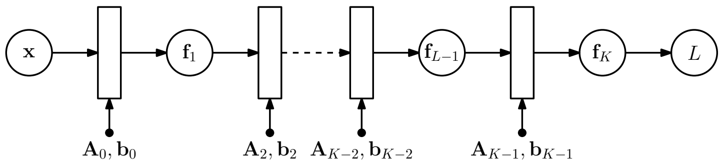


Figure 5.6 Forward pass in a multi-layer neural network to compute the loss function L as a function of the inputs $\boldsymbol{x}$ and the parameters of the network.

we compute the gradient

$$\begin{aligned}\frac{df}{dx} &= -\frac{2x + 2x\exp(x^2)}{2\sqrt{x^2+\exp(x^2)}} - \sin\left(x^2+\exp(x^2)\right)\left(2x+2x\exp(x^2)\right)\\ &= -2x\left(\frac{1}{2\sqrt{x^2+\exp(x^2)}} + \sin\left(x^2+\exp(x^2)\right)\right)\left(1+\exp(x^2)\right).\end{aligned} \tag{5.117}$$

Writing out the gradient in this explicit way is often impractical since it often results in a very lengthy expression for a derivative. In practice, it means that, if we are not careful, the implementation of the gradient could be significantly more expensive than computing the function, which is an unnecessary overhead. For training deep neural network models, the *backpropagation* algorithm (Kelley, 1960; Bryson, 1961; Dreyfus, 1962; Rumelhart et al., 1986) is an efficient way to compute the gradient of an error function with respect to the parameters of the model.

backpropagation

5.6.1 Gradients in a Deep Network

In machine learning, the chain rule plays an important role when optimizing parameters of a hierarchical model (e.g., for maximum likelihood estimation). An area where the chain rule is used to an extreme is Deep Learning where the function value $\boldsymbol{y}$ is computed as a deep function composition

$$\boldsymbol{y} = (f_K \circ f_{K-1} \circ \cdots \circ f_1)(\boldsymbol{x}) = f_K(f_{K-1}(\cdots(f_1(\boldsymbol{x}))\cdots)), \tag{5.118}$$

where $\boldsymbol{x}$ are the inputs (e.g., images), $\boldsymbol{y}$ are the observations (e.g., class labels) and every function f_i, $i = 1, \ldots, K$ possesses its own parameters. In neural networks with multiple layers, we have functions $f_i(\boldsymbol{x}) = \sigma(\boldsymbol{A}_i\boldsymbol{x}_{i-1} + \boldsymbol{b}_i)$ in the ith layer. Here $\boldsymbol{x}_{i-1}$ is the output of layer $i-1$ and σ an activation function, such as the logistic sigmoid $\frac{1}{1+e^{-x}}$, $\tanh$ or a rectified linear unit (ReLU). In order to train these models, we require the gradient of a loss function L with respect to all model parameters $\boldsymbol{A}_j, \boldsymbol{b}_j$ for $i = 0, \ldots, K-1$. This also requires us to compute the gradient of L with respect to the inputs of each layer.

We discuss the case where the activation functions are identical to clutter notation.

For example, if we have inputs $\boldsymbol{x}$ and observations $\boldsymbol{y}$ and a network structure defined by

$$\boldsymbol{f}_0 = \boldsymbol{x} \tag{5.119}$$

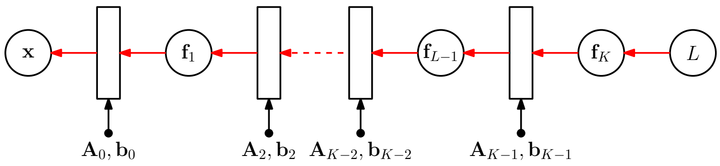


Figure 5.7 Backward pass in a multi-layer neural network to compute the gradients of the loss function.

$$\boldsymbol{f}_i = \sigma_i(\boldsymbol{A}_{i-1}\boldsymbol{f}_{i-1} + \boldsymbol{b}_{i-1})\,, \quad i = 1, \ldots, K\,, \tag{5.120}$$

see also Figure 5.6 for a visualization, we may be interested in finding $\boldsymbol{A}_j, \boldsymbol{b}_j$ for $j = 0, \ldots, K-1$, such that the squared loss

$$L(\boldsymbol{\theta}) = \|\boldsymbol{y} - \boldsymbol{f}_K(\boldsymbol{\theta}, \boldsymbol{x})\|^2 \tag{5.121}$$

is minimized, where $\boldsymbol{\theta} = \{\boldsymbol{A}_j, \boldsymbol{b}_j\}$, $j = 0, \ldots, K-1$.

To obtain the gradients with respect to the parameter set $\boldsymbol{\theta}$, we require the partial derivatives of L with respect to the parameters $\boldsymbol{\theta}_j = \{\boldsymbol{A}_j, \boldsymbol{b}_j\}$ of each layer $j = 0, \ldots, K-1$. The chain rule allows us to determine the partial derivatives as

$$\frac{\partial L}{\partial \boldsymbol{\theta}_{K-1}} = \frac{\partial L}{\partial \boldsymbol{f}_K} \frac{\partial \boldsymbol{f}_K}{\partial \boldsymbol{\theta}_{K-1}} \tag{5.122}$$

$$\frac{\partial L}{\partial \boldsymbol{\theta}_{K-2}} = \frac{\partial L}{\partial \boldsymbol{f}_K} \boxed{\frac{\partial \boldsymbol{f}_K}{\partial \boldsymbol{f}_{K-1}} \frac{\partial \boldsymbol{f}_{K-1}}{\partial \boldsymbol{\theta}_{K-2}}} \tag{5.123}$$

$$\frac{\partial L}{\partial \boldsymbol{\theta}_{K-3}} = \frac{\partial L}{\partial \boldsymbol{f}_K} \frac{\partial \boldsymbol{f}_K}{\partial \boldsymbol{f}_{K-1}} \boxed{\frac{\partial \boldsymbol{f}_{K-1}}{\partial \boldsymbol{f}_{K-2}} \frac{\partial \boldsymbol{f}_{K-2}}{\partial \boldsymbol{\theta}_{K-3}}} \tag{5.124}$$

$$\frac{\partial L}{\partial \boldsymbol{\theta}_i} = \frac{\partial L}{\partial \boldsymbol{f}_K} \frac{\partial \boldsymbol{f}_K}{\partial \boldsymbol{f}_{K-1}} \cdots \boxed{\frac{\partial \boldsymbol{f}_{i+2}}{\partial \boldsymbol{f}_{i+1}} \frac{\partial \boldsymbol{f}_{i+1}}{\partial \boldsymbol{\theta}_i}} \tag{5.125}$$

The **orange** terms are partial derivatives of the output of a layer with respect to its inputs, whereas the **blue** terms are partial derivatives of the output of a layer with respect to its parameters. Assuming, we have already computed the partial derivatives $\partial L/\partial \boldsymbol{\theta}_{i+1}$, then most of the computation can be reused to compute $\partial L/\partial \boldsymbol{\theta}_i$. The additional terms that we need to compute are indicated by the boxes. Figure 5.7 visualizes that the gradients are passed backward through the network. A more in-depth discussion about gradients of neural networks can be found at `https://tinyurl.com/yalcxgtv`.

There are efficient ways of implementing this repeated application of the chain rule using *backpropagation* (Kelley, 1960; Bryson, 1961; Dreyfus, 1962; Rumelhart et al., 1986). A good discussion about backpropagation and the chain rule is available at `https://tinyurl.com/ycfm2yrw`.

backpropagation

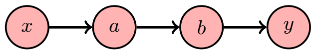


Figure 5.8 Simple graph illustrating the flow of data from x to y via some intermediate variables a, b.

5.6.2 Automatic Differentiation

automatic differentiation

Automatic differentiation is different from symbolic differentiation and numerical approximations of the gradient, e.g., by using finite differences.

It turns out that backpropagation is a special case of a general technique in numerical analysis called *automatic differentiation*. We can think of automatic differentation as a set of techniques to numerically (in contrast to symbolically) evaluate the exact (up to machine precision) gradient of a function by working with intermediate variables and applying the chain rule. Automatic differentiation applies a series of elementary arithmetic operations, e.g., addition and multiplication and elementary functions, e.g., $\sin, \cos, \exp, \log$. By applying the chain rule to these operations, the gradient of quite complicated functions can be computed automatically. Automatic differentiation applies to general computer programs and has forward and reverse modes.

Figure 5.8 shows a simple graph representing the data flow from inputs x to outputs y via some intermediate variables a, b. If we were to compute the derivative dy/dx, we would apply the chain rule and obtain

$$\frac{dy}{dx} = \frac{dy}{db}\frac{db}{da}\frac{da}{dx}\,. \tag{5.126}$$

In the general case, we work with Jacobians, which can be vectors, matrices or tensors.

Intuitively, the forward and reverse mode differ in the order of multiplication. Due to the associativity of matrix multiplication we can choose between

$$\frac{dy}{dx} = \left(\frac{dy}{db}\frac{db}{da}\right)\frac{da}{dx}\,, \tag{5.127}$$

$$\frac{dy}{dx} = \frac{dy}{db}\left(\frac{db}{da}\frac{da}{dx}\right)\,. \tag{5.128}$$

reverse mode

forward mode

Equation (5.127) would be the *reverse mode* because gradients are propagated backward through the graph, i.e., reverse to the data flow. Equation (5.128) would be the *forward mode*, where the gradients flow with the data from left to right through the graph.

In the following, we will focus on reverse mode automatic differentiation, which is backpropagation. In the context of neural networks, where the input dimensionality is often much higher than the dimensionality of the labels, the reverse mode is computationally significantly cheaper than the forward mode. Let us start with an instructive example.

Example

Consider the function

$$f(x) = \sqrt{x^2 + \exp(x^2)} + \cos\left(x^2 + \exp(x^2)\right) \tag{5.129}$$

intermediate variables

from (5.116). If we were to implement a function f on a computer, we would be able to save some computation by using *intermediate variables*:

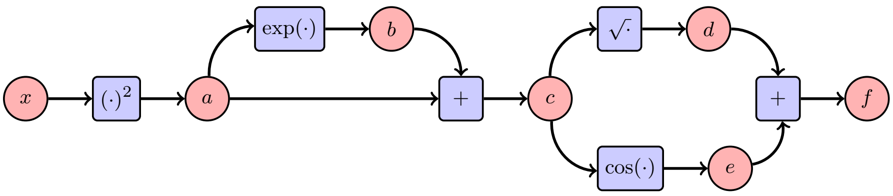


Figure 5.9 Computation graph with inputs x, function values f and intermediate variables a, b, c, d, e.

$$a = x^2\,, \tag{5.130}$$
$$b = \exp(a)\,, \tag{5.131}$$
$$c = a + b\,, \tag{5.132}$$
$$d = \sqrt{c}\,, \tag{5.133}$$
$$e = \cos(c)\,, \tag{5.134}$$
$$f = d + e\,. \tag{5.135}$$

This is the same kind of thinking process that occurs when applying the chain rule. Observe that the above set of equations require fewer operations than a direct naive implementation of the function $f(x)$ as defined in (5.116). The corresponding computation graph in Figure 5.9 shows the flow of data and computations required to obtain the function value f.

The set of equations that include intermediate variables can be thought of as a computation graph, a representation that is widely used in implementations of neural network software libraries. We can directly compute the derivatives of the intermediate variables with respect to their corresponding inputs by recalling the definition of the derivative of elementary functions. We obtain:

$$\frac{\partial a}{\partial x} = 2x\,, \tag{5.136}$$
$$\frac{\partial b}{\partial a} = \exp(a)\,, \tag{5.137}$$
$$\frac{\partial c}{\partial a} = 1 = \frac{\partial c}{\partial b}\,, \tag{5.138}$$
$$\frac{\partial d}{\partial c} = -\frac{1}{2\sqrt{c}}\,, \tag{5.139}$$
$$\frac{\partial e}{\partial c} = -\sin(c)\,, \tag{5.140}$$
$$\frac{\partial f}{\partial d} = 1 = \frac{\partial f}{\partial e}\,. \tag{5.141}$$

By looking at the computation graph in Figure 5.9, we can compute $\partial f/\partial x$ by working backward from the output, and we obtain the following relations:

$$\frac{\partial f}{\partial c} = \frac{\partial f}{\partial d}\frac{\partial d}{\partial c} + \frac{\partial f}{\partial e}\frac{\partial d}{\partial c}\,, \tag{5.142}$$

$$\frac{\partial f}{\partial b} = \frac{\partial f}{\partial c}\frac{\partial c}{\partial b}\,, \tag{5.143}$$
$$\frac{\partial f}{\partial a} = \frac{\partial f}{\partial b}\frac{\partial b}{\partial a} + \frac{\partial f}{\partial c}\frac{\partial c}{\partial a}\,, \tag{5.144}$$
$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial a}\frac{\partial a}{\partial x}\,. \tag{5.145}$$

Note that we have implicitly applied the chain rule to obtain $\partial f/\partial x$. By substituting the results of the derivatives of the elementary functions, we get

$$\frac{\partial f}{\partial c} = 1 \cdot \left(-\frac{1}{2\sqrt{c}}\right) + 1 \cdot (-\sin(c))\ , \tag{5.146}$$
$$\frac{\partial f}{\partial b} = \frac{\partial f}{\partial c} \cdot 1\,, \tag{5.147}$$
$$\frac{\partial f}{\partial a} = \frac{\partial f}{\partial b}\exp(a) + \frac{\partial f}{\partial c} \cdot 1\,, \tag{5.148}$$
$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial a} \cdot 2x\,. \tag{5.149}$$

By thinking of each of the derivatives above as a variable, we observe that the computation required for calculating the derivative is of similar complexity as the computation of the function itself. This is quite counter-intuitive since the mathematical expression for the derivative $\frac{\partial f}{\partial x}$ (5.117) is significantly more complicated than the mathematical expression of the function $f(x)$ in (5.116).

Automatic differentiation is a formalization of the example above. Let $x_1, \ldots, x_d$ be the input variables to the function, $x_{d+1}, \ldots, x_{D-1}$ be the intermediate variables and x_D the output variable. Then the computation graph can be expressed as an equation

$$\text{For } i = d+1, \ldots, D: \quad x_i = g_i(x_{\mathrm{Pa}(x_i)}) \tag{5.150}$$

where $g_i(\cdot)$ are elementary functions and $x_{\mathrm{Pa}(x_i)}$ are the parent nodes of the variable x_i in the graph. Given a function defined in this way, we can use the chain rule to compute the derivative of the function in a step-by-step fashion. Recall that by definition $f = x_D$ and hence

$$\frac{\partial f}{\partial x_D} = 1. \tag{5.151}$$

For other variables x_i, we apply the chain rule

$$\frac{\partial f}{\partial x_i} = \sum_{x_j : x_i \in \mathrm{Pa}(x_j)} \frac{\partial f}{\partial x_j}\frac{\partial x_j}{\partial x_i} = \sum_{x_j : x_i \in \mathrm{Pa}(x_j)} \frac{\partial f}{\partial x_j}\frac{\partial g_j}{\partial x_i}\,, \tag{5.152}$$

where $\mathrm{Pa}(x_j)$ is the set of parent nodes of x_j in the computing graph. Equation (5.150) is the forward propagation of a function, whereas (5.152)

Auto-differentiation in reverse mode requires a parse tree.

is the backpropagation of the gradient through the computation graph. For neural network training we backpropagate the error of the prediction with respect to the label.

The automatic differentiation approach above works whenever we have a function that can be expressed as a computation graph, where the elementary functions are differentiable. In fact, the function may not even be a mathematical function but a computer program. However, not all computer programs can be automatically differentiated, e.g., if we cannot find differential elementary functions. Programming structures, such as `for` loops and `if` statements require more care as well.

5.7 Higher-order Derivatives

So far, we discussed gradients, i.e., first-order derivatives. Sometimes, we are interested in derivatives of higher order, e.g., when we want to use Newton's Method for optimization, which requires second-order derivatives (Nocedal and Wright, 2006). In Section 5.1.1, we discussed the Taylor series to approximate functions using polynomials. In the multivariate case, we can do exactly the same. In the following, we will do exactly this. But let us start with some notation.

Consider a function $f : \mathbb{R}^2 \to \mathbb{R}$ of two variables x, y. We use the following notation for higher-order partial derivatives (and for gradients):

- $\frac{\partial^2 f}{\partial x^2}$ is the second partial derivative of f with respect to x
- $\frac{\partial^n f}{\partial x^n}$ is the nth partial derivative of f with respect to x
- $\frac{\partial^2 f}{\partial x \partial y}$ is the partial derivative obtained by first partial differentiating by x and then y
- $\frac{\partial^2 f}{\partial y \partial x}$ is the partial derivative obtained by first partial differentiating by y and then x

The *Hessian* is the collection of all second-order partial derivatives.

Hessian

If $f(x, y)$ is a twice (continuously) differentiable function then

$$\frac{\partial^2 f}{\partial x \partial y} = \frac{\partial^2 f}{\partial y \partial x}, \tag{5.153}$$

i.e., the order of differentiation does not matter, and the corresponding *Hessian matrix*

Hessian matrix

$$\boldsymbol{H} = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial y^2} \end{bmatrix} \tag{5.154}$$

is symmetric. Generally, for $\boldsymbol{x} \in \mathbb{R}^n$ and $f : \mathbb{R}^n \to \mathbb{R}$, the Hessian is an $n \times n$ matrix. The Hessian measures the local geometry of curvature.

Remark (Hessian of a Vector Field). If $f : \mathbb{R}^n \to \mathbb{R}^m$ is a vector field, the Hessian is an $(m \times n \times n)$-tensor. $\diamondsuit$

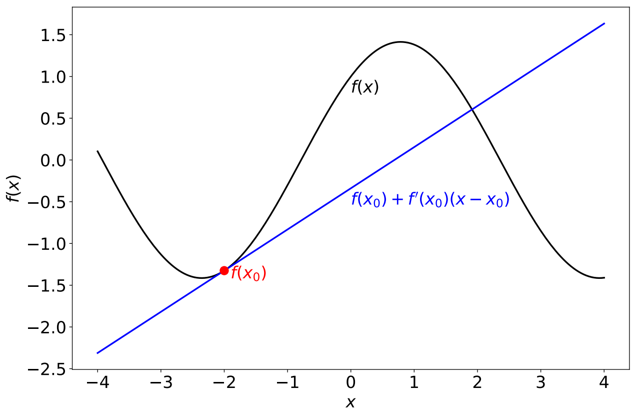


Figure 5.10 Linear approximation of a function. The original function f is linearized at $x_0 = -2$ using a first-order Taylor series expansion.

5.8 Linearization and Multivariate Taylor Series

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

$$f(\boldsymbol{x}) \approx f(\boldsymbol{x}_0) + (\nabla_{\boldsymbol{x}} f)(\boldsymbol{x}_0)(\boldsymbol{x} - \boldsymbol{x}_0)\,. \tag{5.155}$$

Here $(\nabla_{\boldsymbol{x}} f)(\boldsymbol{x}_0)$ is the gradient of f with respect to $\boldsymbol{x}$, evaluated at $\boldsymbol{x}_0$. Figure 5.10 illustrates the linear approximation of a function f at an input x_0. The orginal function is approximated by a straight line. This approximation is locally accurate, but the further we move away from x_0 the worse the approximation gets. Equation (5.155) is a special case of a multivariate Taylor series expansion of f at $\boldsymbol{x}_0$, where we consider only the first two terms. We discuss the more general case in the following, which will allow for better approximations.

multivariate Taylor series

Definition 5.7 (Multivariate Taylor Series)**.** For the *multivariate Taylor series*, we consider a function

$$f : \mathbb{R}^D \to \mathbb{R} \tag{5.156}$$

$$\boldsymbol{x} \mapsto f(\boldsymbol{x})\,, \quad \boldsymbol{x} \in \mathbb{R}^D\,, \tag{5.157}$$

that is smooth at $\boldsymbol{x}_0$.

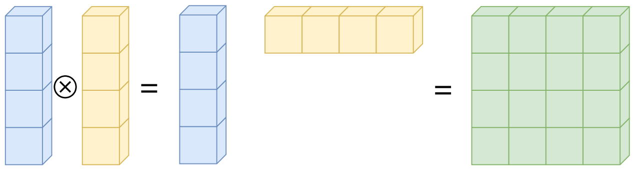
When we define the difference vector $\boldsymbol{\delta} := \boldsymbol{x} - \boldsymbol{x}_0$, the Taylor series of f at $(\boldsymbol{x}_0)$ is defined as

$$f(\boldsymbol{x}) = \sum_{k=0}^{\infty} \frac{D_{\boldsymbol{x}}^k f(\boldsymbol{x}_0)}{k!} \boldsymbol{\delta}^k\,, \tag{5.158}$$

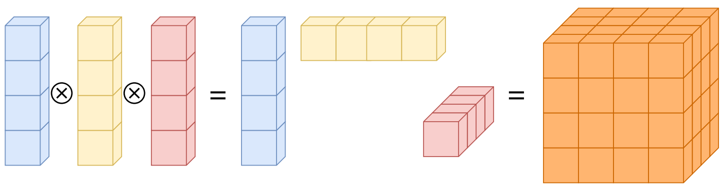
where $D_{\boldsymbol{x}}^k f(\boldsymbol{x}_0)$ is the k-th (total) derivative of f with respect to $\boldsymbol{x}$, evaluated at $\boldsymbol{x}_0$.

Taylor polynomial

Definition 5.8 (Taylor Polynomial)**.** The *Taylor polynomial* of degree n of f at $\boldsymbol{x}_0$ contains the first $n+1$ components of the series in (5.158) and is



(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.

Figure 5.11 Visualizing outer products. Outer products of vectors increase the dimensionality of the array by 1 per term.

defined as

$$T_n = \sum_{k=0}^{n} \frac{D_{\boldsymbol{x}}^k f(\boldsymbol{x}_0)}{k!} \boldsymbol{\delta}^k \,. \tag{5.159}$$

Remark (Notation). In (5.158) and (5.159), we used the slightly sloppy notation of $\boldsymbol{\delta}^k$, which is not defined for vectors $\boldsymbol{x} \in \mathbb{R}^D$, $D > 1$, and $k > 1$. Note that both $D_x^k f$ and $\boldsymbol{\delta}^k$ are k-th order tensors, i.e., k-dimensional arrays. The k-th order tensor $\boldsymbol{\delta}^k \in \mathbb{R}^{\overbrace{D\times D\times\ldots\times D}^{k \text{ times}}}$ is obtained as a k-fold outer product, denoted by $\otimes$, of the vector $\boldsymbol{\delta} \in \mathbb{R}^D$. For example,

A vector can be implemented as a 1-dimensional array, a matrix as a 2-dimensional array.

$$\boldsymbol{\delta}^2 = \boldsymbol{\delta} \otimes \boldsymbol{\delta} = \boldsymbol{\delta}\boldsymbol{\delta}^\top \,, \quad \boldsymbol{\delta}^2[i,j] = \delta[i]\delta[j] \tag{5.160}$$

$$\boldsymbol{\delta}^3 = \boldsymbol{\delta} \otimes \boldsymbol{\delta} \otimes \boldsymbol{\delta} \,, \quad \boldsymbol{\delta}^3[i,j,k] = \delta[i]\delta[j]\delta[k] \,. \tag{5.161}$$

Figure 5.11 visualizes two such outer products. In general, we obtain the following terms in the Taylor series:

$$D_x^k f(\boldsymbol{x}_0)\boldsymbol{\delta}^k = \sum_a \cdots \sum_k D_x^k f(\boldsymbol{x}_0)[a,\ldots,k]\delta[a]\cdots\delta[k] \,, \tag{5.162}$$

3012 where $D_x^k f(\boldsymbol{x}_0)\boldsymbol{\delta}^k$ contains k-th order polynomials.

Now that we defined the Taylor series for vector fields, let us explicitly write down the first terms $D_x^k f(\boldsymbol{x}_0)\boldsymbol{\delta}^k$ of the Taylor series expansion for $k = 0, \ldots, 3$ and $\boldsymbol{\delta} := \boldsymbol{x} - \boldsymbol{x}_0$:

```
np.einsum('i,i',Df1,d)
np.einsum('ij,i,j',Df2,d,d)
np.einsum('ijk,i,j,k',Df3,d,d,d)
```

$$k = 0 : D_x^0 f(\boldsymbol{x}_0)\boldsymbol{\delta}^0 = f(\boldsymbol{x}_0) \in \mathbb{R} \tag{5.163}$$

$$k = 1 : D_x^1 f(\boldsymbol{x}_0)\boldsymbol{\delta}^1 = \underbrace{\nabla_{\boldsymbol{x}} f(\boldsymbol{x}_0)}_{1\times D} \underbrace{\boldsymbol{\delta}}_{D\times 1} = \sum_i \nabla_{\boldsymbol{x}} f(\boldsymbol{x}_0)[i]\delta[i] \in \mathbb{R} \tag{5.164}$$

$$k = 2 : D_x^2 f(\boldsymbol{x}_0)\boldsymbol{\delta}^2 = \operatorname{tr}\big(\underbrace{\boldsymbol{H}}_{D\times D} \underbrace{\boldsymbol{\delta}}_{D\times 1} \underbrace{\boldsymbol{\delta}^\top}_{1\times D}\big) = \boldsymbol{\delta}^\top \boldsymbol{H}\boldsymbol{\delta} \tag{5.165}$$

$$= \sum_i \sum_j H[i,j]\delta[i]\delta[j] \in \mathbb{R} \tag{5.166}$$

$$k = 3 : D_x^3 f(\boldsymbol{x}_0)\boldsymbol{\delta}^3 = \sum_i \sum_j \sum_k D_x^3 f(\boldsymbol{x}_0)[i,j,k]\delta[i]\delta[j]\delta[k] \in \mathbb{R} \tag{5.167}$$

$\diamondsuit$

Example (Taylor-Series Expansion of a Function with Two Variables)
Consider the function

$$f(x,y) = x^2 + 2xy + y^3\,. \tag{5.168}$$

We want to compute the Taylor series expansion of f at $(x_0, y_0) = (1, 2)$. Before we start, let us discuss what to expect: The function in (5.168) is a polynomial of degree 3. We are looking for a Taylor series expansion, which itself is a linear combination of polynomials. Therefore, we do not expect the Taylor series expansion to contain terms of fourth or higher order to express a third-order polynomial. This means, it should be sufficient to determine the first four terms of (5.158) for an exact alternative representation of (5.168).

To determine the Taylor series expansion, start of with the constant term and the first-order derivatives, which are given by

$$f(1,2) = 13 \tag{5.169}$$

$$\frac{\partial f}{\partial x} = 2x + 2y \implies \frac{\partial f}{\partial x}(1,2) = 6 \tag{5.170}$$

$$\frac{\partial f}{\partial y} = 2x + 3y^2 \implies \frac{\partial f}{\partial y}(1,2) = 14\,. \tag{5.171}$$

Therefore, we obtain

$$D^1_{x,y} f(1,2) = \nabla_{x,y} f(1,2) = \begin{bmatrix} \frac{\partial f}{\partial x}(1,2) & \frac{\partial f}{\partial y}(1,2) \end{bmatrix} = \begin{bmatrix} 6 & 14 \end{bmatrix} \in \mathbb{R}^{1\times 2} \tag{5.172}$$

such that

$$\frac{D^1_{x,y} f(1,2)}{1!}\boldsymbol{\delta} = \begin{bmatrix} 6 & 14 \end{bmatrix} \begin{bmatrix} x-1 \\ y-2 \end{bmatrix} = 6(x-1) + 14(y-2)\,. \tag{5.173}$$

Note that $D^1_{x,y} f(1,2)\boldsymbol{\delta}$ contains only linear terms, i.e., first-order polynomials.

The second-order partial derivatives are given by

$$\frac{\partial^2 f}{\partial x^2} = 2 \implies \frac{\partial^2 f}{\partial x^2}(1,2) = 2 \tag{5.174}$$

$$\frac{\partial^2 f}{\partial y^2} = 6y \implies \frac{\partial^2 f}{\partial y^2}(1,2) = 12 \tag{5.175}$$

$$\frac{\partial f^2}{\partial x \partial y} = 2 \implies \frac{\partial f^2}{\partial x \partial y}(1,2) = 2 \tag{5.176}$$

$$\frac{\partial f^2}{\partial y \partial x} = 2 \implies \frac{\partial f^2}{\partial y \partial x}(1,2) = 2\,. \tag{5.177}$$

When we collect the second-order partial derivatives, we obtain the Hessian

$$\boldsymbol{H} = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial f^2}{\partial x \partial y} \\ \frac{\partial f^2}{\partial y \partial x} & \frac{\partial^2 f}{\partial y^2} \end{bmatrix} = \begin{bmatrix} 2 & 2 \\ 2 & 6y \end{bmatrix}, \tag{5.178}$$

such that

$$\boldsymbol{H}(1,2) = \begin{bmatrix} 2 & 2 \\ 2 & 12 \end{bmatrix} \in \mathbb{R}^{2\times 2}\,. \tag{5.179}$$

Therefore, the next term of the Taylor-series expansion is given by

$$\frac{D^2_{x,y} f(1,2)}{2!} \boldsymbol{\delta}^2 = \frac{1}{2} \boldsymbol{\delta}^\top \boldsymbol{H}(1,2) \boldsymbol{\delta} \tag{5.180}$$

$$= \begin{bmatrix} x-1 & y-2 \end{bmatrix} \begin{bmatrix} 2 & 2 \\ 2 & 12 \end{bmatrix} \begin{bmatrix} x-1 \\ y-2 \end{bmatrix} \tag{5.181}$$

$$= (x-1)^2 + 2(x-1)(y-2) + 6(y-2)^2\,. \tag{5.182}$$

Here, $D^2_{x,y} f(1,2)\boldsymbol{\delta}^2$ contains only quadratic terms, i.e., second-order polynomials.

The third-order derivatives are obtained as

$$D^3_x f = \begin{bmatrix} \frac{\partial \boldsymbol{H}}{\partial x} & \frac{\partial \boldsymbol{H}}{\partial y} \end{bmatrix} \in \mathbb{R}^{2\times 2\times 2}, \tag{5.183}$$

$$D^3_x f[:,:,1] = \frac{\partial \boldsymbol{H}}{\partial x} = \begin{bmatrix} \frac{\partial^3 f}{\partial x^3} & \frac{\partial^3 f}{\partial x \partial y \partial x} \\ \frac{\partial^3 f}{\partial y \partial x^2} & \frac{\partial^3 f}{\partial y^2 \partial x} \end{bmatrix}, \tag{5.184}$$

$$D^3_x f[:,:,2] = \frac{\partial \boldsymbol{H}}{\partial y} = \begin{bmatrix} \frac{\partial^3 f}{\partial x^2 \partial y} & \frac{\partial^3 f}{\partial x \partial y^2} \\ \frac{\partial^3 f}{\partial y \partial x \partial y} & \frac{\partial^3 f}{\partial y^3}\,. \end{bmatrix}. \tag{5.185}$$

Since most second-order partial derivatives in the Hessian in (5.178) are constant the only non-zero third-order partial derivative is

$$\frac{\partial^3 f}{\partial y^3} = 6 \implies \frac{\partial^3 f}{\partial y^3}(1,2) = 6\,. \tag{5.186}$$

Higher-order derivatives and the mixed derivatives of degree 3 (e.g., $\frac{\partial f^3}{\partial x^2 \partial y}$) vanish, such that

$$D^3_{x,y} f[:,:,1] = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, \quad D^3_{x,y} f[:,:,2] = \begin{bmatrix} 0 & 0 \\ 0 & 6 \end{bmatrix} \tag{5.187}$$

and

$$\frac{D^3_{x,y} f(1,2)}{3!} \boldsymbol{\delta}^3 = (y-2)^3\,, \tag{5.188}$$

which collects all cubic terms (third-order polynomials) of the Taylor series.

Overall, the (exact) Taylor series expansion of f at $(x_0, y_0) = (1, 2)$ is

$$
\begin{aligned}
f(x) &= f(1,2) + D^1_{x,y} f(1,2)\boldsymbol{\delta} + \frac{D^2_{x,y} f(1,2)}{2!}\boldsymbol{\delta}^2 + \frac{D^3_{x,y} f(1,2)}{3!}\boldsymbol{\delta}^3 && (5.189)\\
&= f(1,2) + \frac{\partial f(1,2)}{\partial x}(x-1) + \frac{\partial f(1,2)}{\partial y}(y-2) && (5.190)\\
&\quad + \frac{1}{2!}\left(\frac{\partial^2 f(1,2)}{\partial x^2}(x-1)^2 + \frac{\partial^2 f(1,2)}{\partial y^2}(y-2)^2\right. && (5.191)\\
&\quad \left. + 2\frac{\partial^2 f(1,2)}{\partial x \partial y}(x-1)(y-2)\right) + \frac{1}{6}\frac{\partial^3 f(1,2)}{\partial y^3}(y-2)^3 && (5.192)\\
&= 13 + 6(x-1) + 14(y-2) && (5.193)\\
&\quad + (x-1)^2 + 6(y-2)^2 + 2(x-1)(y-2) + (y-2)^3 \,. && (5.194)
\end{aligned}
$$

In this case, we obtained an exact Taylor series expansion of the polynomial in (5.168), i.e., the polynomial in (5.194) is identical to the original polynomial in (5.168). In this particular example, this result is not surprising since the original function was a third-order polynomial, which we expressed through a linear combination of constant terms, first-order, second order and third-order polynomials in (5.194).

5.9 Further Reading

In machine learning (and other disciplines), we often need to compute expectations, i.e., we need to solve integrals of the form

$$\mathbb{E}_{\boldsymbol{x}}[f(\boldsymbol{x})] = \int f(\boldsymbol{x})p(\boldsymbol{x})d\boldsymbol{x} \,. \tag{5.195}$$

Extended Kalman Filter

unscented transform

Laplace approximation

Even if $p(\boldsymbol{x})$ is in a convenient form (e.g., Gaussian), this integral generally not be solved analytically. The Taylor series expansion of f is one way of finding an approximate solution: Assuming $p(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ is Gaussian, then the first-order Taylor series expansion around $\boldsymbol{\mu}$ locally linearizes the nonlinear function f. For linear functions, we can compute the mean (and the covariance) exactly if $p(\boldsymbol{x})$ is Gaussian distributed (see Section 6.5). This property is heavily exploited by the *Extended Kalman Filter* (Maybeck, 1979) for online state estimation in nonlinear dynamical systems (also called "state-space models"). Other deterministic ways to approximate the integral in (5.195) are the *unscented transform* (Julier and Uhlmann, 1997), which does not require any gradients, or the *Laplace approximation* (Bishop, 2006), which uses the Hessian for a local Gaussian approximation of $p(\boldsymbol{x})$ at the posterior mean.

Exercises

5.1 Consider the following functions

$$f_1(\boldsymbol{x}) = \sin(x_1)\cos(x_2)\,, \quad \boldsymbol{x} \in \mathbb{R}^2 \tag{5.196}$$

$$f_2(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{x}^\top \boldsymbol{y}\,, \quad \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n \tag{5.197}$$

$$f_3(\boldsymbol{x}) = \boldsymbol{x}\boldsymbol{x}^\top\,, \qquad \boldsymbol{x} \in \mathbb{R}^n \tag{5.198}$$

1. What are the dimensions of $\frac{\partial f_i}{\partial \boldsymbol{x}}$?
2. Compute the Jacobians

5.2 Differentiate f with respect to $\boldsymbol{t}$ and g with respect to $\boldsymbol{X}$, where

$$f(\boldsymbol{t}) = \sin(\log(\boldsymbol{t}^\top \boldsymbol{t}))\,, \qquad t \in \mathbb{R}^D \tag{5.199}$$

$$g(\boldsymbol{X}) = \operatorname{tr}(\boldsymbol{A}\boldsymbol{X}\boldsymbol{B})\,, \qquad \boldsymbol{A} \in \mathbb{R}^{D\times E}, \boldsymbol{X} \in \mathbb{R}^{E\times F}, \boldsymbol{B} \in \mathbb{R}^{F\times D}\,, \tag{5.200}$$

where tr denotes the trace.

5.3 Compute the derivatives $df/d\boldsymbol{x}$ of the following functions by using the chain rule. Provide the dimensions of every single partial derivative. Describe your steps in detail.

1.

$$f(z) = \log(1+z)\,, \quad z = \boldsymbol{x}^\top \boldsymbol{x}\,, \quad \boldsymbol{x} \in \mathbb{R}^D$$

2.

$$f(\boldsymbol{z}) = \sin(\boldsymbol{z})\,, \quad \boldsymbol{z} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}\,, \quad \boldsymbol{A} \in \mathbb{R}^{E\times D}, \boldsymbol{x} \in \mathbb{R}^D, \boldsymbol{b} \in \mathbb{R}^E$$

where $\sin(\cdot)$ is applied to every element of $\boldsymbol{z}$.

5.4 Compute the derivatives $df/d\boldsymbol{x}$ of the following functions.
Describe your steps in detail.

1. Use the chain rule. Provide the dimensions of every single partial derivative.

$$\begin{aligned} f(z) &= \exp(-\tfrac{1}{2}z) \\ z &= g(\boldsymbol{y}) = \boldsymbol{y}^\top \boldsymbol{S}^{-1}\boldsymbol{y} \\ \boldsymbol{y} &= h(\boldsymbol{x}) = \boldsymbol{x} - \boldsymbol{\mu} \end{aligned}$$

where $\boldsymbol{x}, \boldsymbol{\mu} \in \mathbb{R}^D$, $\boldsymbol{S} \in \mathbb{R}^{D\times D}$.

2.

$$f(\boldsymbol{x}) = \operatorname{tr}(\boldsymbol{x}\boldsymbol{x}^\top + \sigma^2 \boldsymbol{I})\,, \quad \boldsymbol{x} \in \mathbb{R}^D$$

Here $\operatorname{tr}(\boldsymbol{A})$ is the trace of $\boldsymbol{A}$, i.e., the sum of the diagonal elements A_{ii}.
Hint: Explicitly write out the outer product.

3. Use the chain rule. Provide the dimensions of every single partial derivative. You do not need to compute the product of the partial derivatives explicitly.

$$\begin{aligned} \boldsymbol{f} &= \tanh(\boldsymbol{z}) \in \mathbb{R}^M \\ \boldsymbol{z} &= \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}, \quad \boldsymbol{x} \in \mathbb{R}^N, \boldsymbol{A} \in \mathbb{R}^{M\times N}, \boldsymbol{b} \in \mathbb{R}^M. \end{aligned}$$

Here, $\tanh$ is applied to every component of $\boldsymbol{z}$.