1 Some basic math used in this course

1.1 Some linear algebra

We frequently use vector and matrix notation in this book as it is extremely convenient for specifying neural network models. It is a shorthand notation for otherwise lengthy looking formulas, and formulas written in this notation can easily be entered into MATLAB. We consider three basic data types:

1. Scalar:
$$a \text{ for example } 41 \tag{1.1}$$
2. Vector:
$$\mathbf{a} \text{ or component-wise } \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} \text{ for example } \begin{pmatrix} 41 \\ 7 \\ 13 \end{pmatrix} \tag{1.2}$$
3. Matrix:
$$\mathbf{a} \text{ or component-wise } \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{pmatrix} \text{ for example } \begin{pmatrix} 41 & 12 \\ 7 & 45 \\ 13 & 9 \end{pmatrix} \tag{1.3}$$

We used bold face to indicate both a vector and a matrix because the difference is usually apparent from the circumstances. A matrix is just a collection of scalars or vectors. We talk about an $n \times m$ matrix where n is the number of rows and m is the number of columns. A scalar is thus a 1×1 matrix, and a vector of length n can be considered an $n \times 1$ matrix. A similar collection of data is called **array** in computer science. However, a matrix is difference because we also define operations on these data collections. The rules of calculating with matrices can be applied to scalars and vectors.

We define how to add and multiply two matrices so that we can use them in algebraic equations. The **sum of two matrices** is defined as the sum of the individual components
$$(\mathbf{a} + \mathbf{b})_{ij} = \mathbf{a}_{ij} + \mathbf{b}_{ij}. \tag{1.4}$$
For example, $\mathbf{a}$ and $\mathbf{b}$ are 3×2 matrices, then
$$\mathbf{a} + \mathbf{b} = \begin{pmatrix} a_{11} + b_{11} & a_{12} + b_{12} \\ a_{21} + b_{21} & a_{22} + b_{22} \\ a_{31} + b_{31} & a_{32} + b_{32} \end{pmatrix} \tag{1.5}$$
Matrix multiplication is defined as
$$(\mathbf{a} * \mathbf{b})_{ij} = \sum_k \mathbf{a}_{ik} \mathbf{b}_{kj}. \tag{1.6}$$

The matrix multiplication is hence only defined as multiplication matrices $\mathbf{a}$ and $\mathbf{b}$ where the number of columns of the matrix $\mathbf{a}$ is equal to the number of rows of matrix

$\mathbf{b}$. For example, for two square matrices with two rows and two columns, their product is given by

$$\mathbf{a} * \mathbf{b} = \begin{pmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} \end{pmatrix} \tag{1.7}$$

A handy rule for matrix multiplications is illustrated in Fig. 1.1. Each component in the resulting matrix is calculated from the sum of two multiplicative terms. The rule for multiplying two matrices is tedious but straightforward and can easily be implemented in a computer. It is the default when multiplying variables of the matrix type in MATLAB. If we want to multiply each component of a matrix by the corresponding component in a second matrix, we just have to include the operator '`.*`' between the matrices in MATLAB where the dot in front of the multiplication sign indicates 'component-wise'.

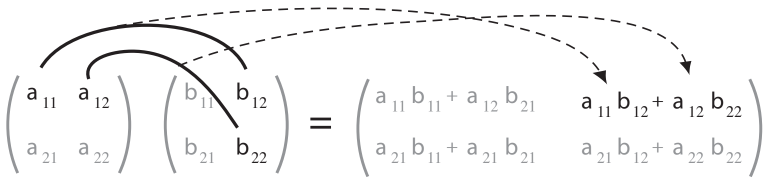


Fig. 1.1 Illustration of a matrix multiplication. Each element in the resulting matrix consists of terms that are taken from the corresponding row of the first matrix and column of the second matrix. Thus in the example we calculate the highlighted element from the components of the first row of the first matrix and the second column of the the second matrix. From these rows and columns we add all the terms that consist of the element-wise multiplication of the terms.

Another useful definition is the **transpose** of a matrix. This operation, indicated usually by a superscript t or a prime ($'$). The later is used in MATLAB. Taking the transpose of a matrix means that the matrix is rotated 90 degrees; the first row becomes the first column, the second row becomes the second column, etc. For example, the transpose of the example in 1.3 is

$$\mathbf{a}' = \begin{pmatrix} 41 & 7 & 13 \\ 12 & 45 & 9 \end{pmatrix} \tag{1.8}$$

The transpose of a vector transforms a column vector into a row vector and vice versa.

As already mentioned, matrices were invented to simplify the notations for systems of coupled algebraic equations. Consider, for example, the system of three equations

$$41x_1 + 12x_2 = 17 \tag{1.9}$$
$$7x_1 + 45x_2 = -83 \tag{1.10}$$
$$13x_1 + 9x_2 = -5. \tag{1.11}$$

This can be written as

$$\mathbf{ax} = \mathbf{b} \tag{1.12}$$

with the matrix $\mathbf{a}$ as in the example of 1.3, the vector $\mathbf{x} = (x_1 \;\; x_2)'$, and the vector $\mathbf{b} = (17 \;\; -83 \;\; -5)'$.

The solution of this linear equation system is equivalent to finding the inverse of matrix $\mathbf{a}$ which we write as $\mathbf{a}^{-1}$. The inverse of the matrix is defined by

$$\mathbf{a}^{-1}\mathbf{a} = \mathbb{1}, \tag{1.13}$$

where the matrix $\mathbb{1}$ is the unit matrix that has element of one on the diagonal and zeros otherwise. Multiplying equation 1.12 from left with $\mathbf{a}^{-1}$ is hence

$$\mathbf{x} = \mathbf{a}^{-1}\mathbf{b} \tag{1.14}$$

The inverse of a matix, if this exists, can be found by the MATLAB function `inv()`.

1.2 Basic calculus

1.2.1 Differences and sums

We are often interested how a variable change with time. Let us consider the quantity $x(t)$ where we indicated that this quantity depends on time. The change of this variable from time t to time $t' = t + \Delta t$ is then

$$\Delta x = x(t + \Delta t) - x(t). \tag{1.15}$$

The quantity Δt is the finite difference in time. For a continuously changing quantity we could also think about the instantaneous change value, $\mathrm{d}x$, by considering an infinitesimally small time step. Formally,

$$\mathrm{d}x = \lim_{\Delta t \to 0} \Delta x = \lim_{\Delta t \to 0} (x(t + \Delta t) - x(t)). \tag{1.16}$$

The infinitesimally small time step is often written as $\mathrm{d}t$. Calculating with such infinitesimal quantities is covered in the mathematical discipline of calculus, but on the computer we have always finite differences and we need to consider very small time steps to approximate continuous formulation. With discrete time steps, differential become differences and integrals become sums

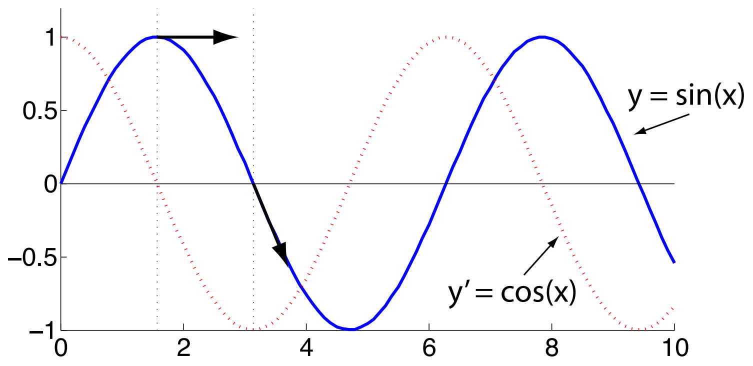
$$\mathrm{d}x \to \Delta x \tag{1.17}$$

$$\int \mathrm{d}x \to \Delta x \sum \tag{1.18}$$

Note the factor of Δx in front of the summation in the last equation. It is easy to forget this factor when replacing integrals with sums.

1.2.2 Derivatives

The derivative of a quantity y that depends on x is the slope of the function $y(x)$. This derivative can be defined as the limiting process equation 1.16 and is commonly written as $\frac{\mathrm{d}y}{\mathrm{d}x}$ or as y'.



It is useful to know some derivatives of basic functions.

$$y = e^x \rightarrow y' = e^x \tag{1.19}$$

$$y = sin(x) \rightarrow y' = cos(x) \tag{1.20}$$

$$y = x^n \rightarrow y' = nx^{n-1} \tag{1.21}$$

$$y = log(x) \rightarrow \frac{1}{x} \tag{1.22}$$

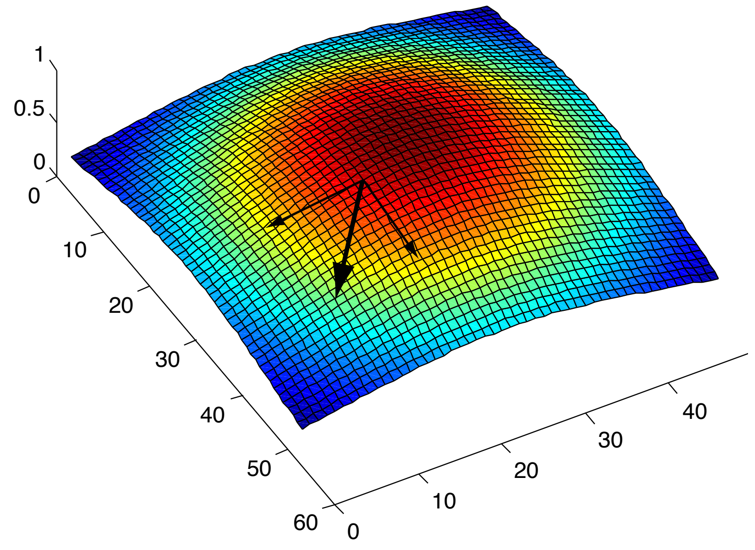
as well as the chain rule

$$y = f(x) \rightarrow y' = \frac{\mathrm{d}y}{\mathrm{d}x} = \frac{\mathrm{d}y}{\mathrm{d}f}\frac{\mathrm{d}f}{\mathrm{d}x}. \tag{1.23}$$

1.2.3 Partial derivative and gradients

A function that depends on more than one variable is a higher dimensional function. An example is the two-dimensional function $z(x, y)$. The slope of the function in the direction x (keeping y constant) is defined as $\frac{\partial z}{\partial x}$ and in the direction of y (keeping x constant) as $\frac{\partial z}{\partial x}$. The **gradient** is the vector that point in the direction of the maximal slope and has a length proportional t the slope,

$$\mathrm{grad} z = \begin{pmatrix} \frac{\partial z}{\partial x} \\ \frac{\partial z}{\partial y} \end{pmatrix}. \tag{1.24}$$


1
0.5
0
0
10
20
30
40
50
60
0
10
20
30
40