The fundamental takeaway is that rows of one matrix are multiplied against columns of the other matrix.

This definition hides a number of subtleties. Note first that matrix multiplication is not commutative. That is, $AB \neq BA$ in general. In fact, AB can exist when BA is not meaningful. Suppose, for example, A is a matrix of shape (2, 3) and B is a matrix of shape (3, 4). Then AB is a matrix of shape (2, 4). However, BA is not defined since the respective dimensions (4 and 2) don't match. As another subtlety, note that, as in the rotation example, a matrix of shape (m, n) can be multiplied on the right by a matrix of shape $(n, 1)$. However, a matrix of shape $(n, 1)$ is simply a column vector. So, it is meaningful to multiply matrices by vectors. Matrix-vector multiplication is one of the fundamental building blocks of common machine learning systems.

One of the nicest properties of standard multiplication is that it is a linear operation. More precisely, a function f is called linear if $f(x + y) = f(x) + f(y)$ and $f(cx) = cf(x)$ where c is a scalar. To demonstrate that scalar multiplication is linear, suppose that a, b, c, d are all real numbers. Then we have

$$a \cdot (b \cdot c) = b \cdot (ac)$$

$$a \cdot (c + d) = ac + ad$$

We make use of the commutative and distributive properties of scalar multiplication here. Now suppose that instead, A, C, D are now matrices where C, D are of the same size and it is meaningful to multiply A on the right with either C or D (b remains a real number). Then matrix multiplication is a linear operator:

$$A(b \cdot C) = b \cdot (AC)$$

$$A(C + D) = AC + AD$$

Put another way, matrix multiplication is distributive and commutes with scalar multiplication. In fact, it can be shown that any linear transformation on vectors corresponds to a matrix multiplication. For a computer science analogy, think of linearity as a property demanded by an abstract method in a superclass. Then standard multiplication and matrix multiplication are concrete implementations of that abstract method for different subclasses (respectively real numbers and matrices).

Tensors

In the previous sections, we introduced the notion of scalars as rank-0 tensors, vectors as rank-1 tensors, and matrices as rank-2 tensors. What then is a rank-3 tensor? Before passing to a general definition, it can help to think about the commonalities

between scalars, vectors, and matrices. Scalars are single numbers. Vectors are lists of numbers. To pick out any particular element of a vector requires knowing its index. Hence, we need one index element into the vector (thus a rank-1 tensor). Matrices are tables of numbers. To pick out any particular element of a matrix requires knowing its row and column. Hence, we need two index elements (thus a rank-2 tensor). It follows naturally that a rank-3 tensor is a set of numbers where there are three required indices. It can help to think of a rank-3 tensor as a rectangular prism of numbers, as illustrated in Figure 2-4.

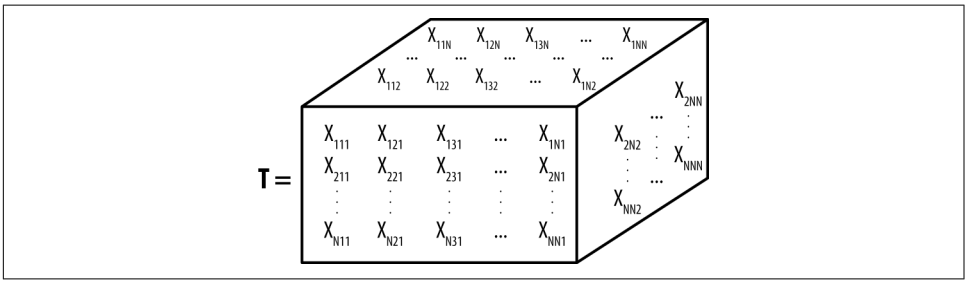

Figure 2-4. A rank-3 tensor can be visualized as a rectangular prism of numbers.

The rank-3 tensor T displayed in the figure is of shape (N, N, N). An arbitrary element of the tensor would then be selected by specifying (i, j, k) as indices.

There is a linkage between tensors and shapes. A rank-1 tensor has a shape of dimension 1, a rank-2 tensor a shape of dimension 2, and a rank-3 tensor of dimension 3. You might protest that this contradicts our earlier discussion of row and column vectors. By our definition, a column vector has shape $(n, 1)$. Wouldn't that make a column vector a rank-2 tensor (or a matrix)? This is exactly what has happened. Recall that a vector which is not specified to be a row vector or column vector has shape (n). When we specify that a vector is a row vector or a column vector, we in fact specify a method of transforming the underlying vector into a matrix. This type of dimension expansion is a common trick in tensor manipulation.

Note that another way of thinking about a rank-3 tensor is as a list of matrices all with the same shape. Suppose that W is a matrix with shape (n, n). Then the tensor $T_{ijk} = (W_1, \cdots, W_n)$ consists of n copies of the matrix W.

Note that a black-and-white image can be represented as a rank-2 tensor. Suppose we have a 224 × 224-pixel black and white image. Then, pixel (i, j) is 1/0 to encode a black/white pixel, respectively. It follows that a black and white image can be represented as a matrix of shape (224, 224). Now, consider a 224 × 224 color image. The color at a particular pixel is typically represented by three separate RGB channels. That is, pixel (i, j) is represented as a tuple of numbers (r, g, b) that encode the amount of red, green, and blue at the pixel, respectively. r, g, b are typically integers from 0 to 255. It follows now that the color image can be encoded as a rank-3 tensor

of shape (224, 224, 3). Continuing the analogy, consider a color video. Suppose that each frame of the video is a 224 × 224 color image. Then a minute of video (at 60 fps) would be a rank-4 tensor of shape (224, 224, 3, 3600). Continuing even further, a collection of 10 such videos would then form a rank-5 tensor of shape (10, 224, 224, 3, 3600). In general, tensors provide for a convenient representation of numeric data. In practice, it's not common to see tensors of higher order than rank-5 tensors, but it's smart to design any tensor software to allow for arbitrary tensors since intelligent users will always come up with use cases designers don't consider.

Tensors in Physics

Tensors are used widely in physics to encode fundamental physical quantities. For example, the stress tensor is commonly used in material science to define the stress at a point within a material. Mathematically, the stress tensor is a rank-2 tensor of shape (3, 3):

$$\sigma = \begin{pmatrix} \sigma_{11} & \tau_{12} & \tau_{13} \\ \tau_{21} & \sigma_{22} & \tau_{23} \\ \tau_{31} & \tau_{32} & \sigma_{33} \end{pmatrix}$$

Then, suppose that n is a vector of shape (3) that encodes a direction. The stress T^n in direction n is specified by the vector $T^n = T \cdot n$ (note the matrix-vector multiplication). This relationship is depicted pictorially in Figure 2-5.

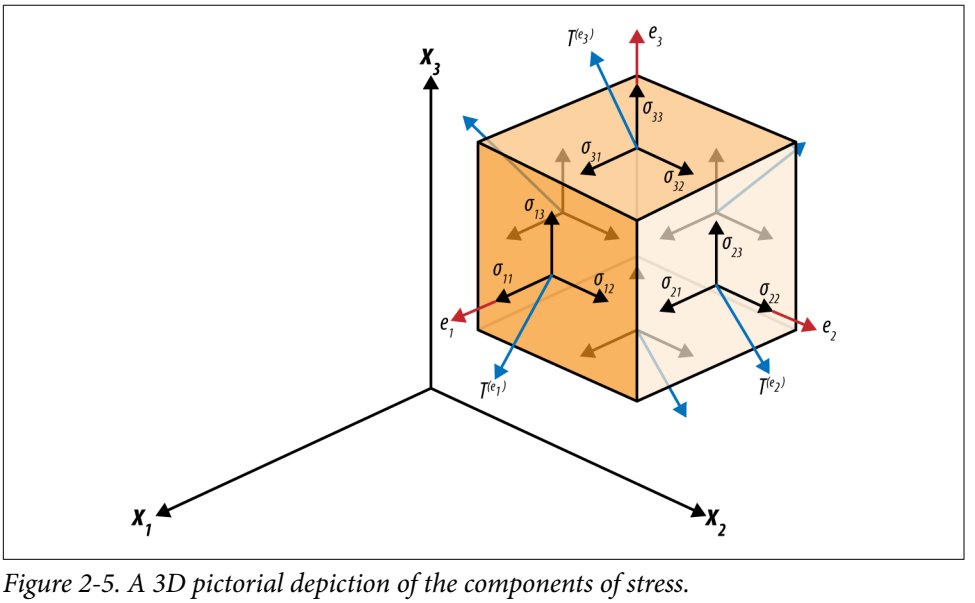

Figure 2-5. A 3D pictorial depiction of the components of stress.