learning and its foundation on continuous, vectorial mathematics. Modern machine learning is founded upon the manipulation and calculus of tensors.

Scalars, Vectors, and Matrices

To start, we will give some simple examples of tensors that you might be familiar with. The simplest example of a tensor is a scalar, a single constant value drawn from the real numbers (recall that the real numbers are decimal numbers of arbitrary precision, with both positive and negative numbers permitted). Mathematically, we denote the real numbers by $\mathbb{R}$. More formally, we call a scalar a rank-0 tensor.

Aside on Fields

Mathematically sophisticated readers will protest that it's entirely meaningful to define tensors based on the complex numbers, or with binary numbers. More generally, it's sufficient that the numbers come from a *field*: a mathematical collection of numbers where 0, 1, addition, multiplication, subtraction, and division are defined. Common fields include the real numbers $\mathbb{R}$, the rational numbers $\mathbb{Q}$, the complex numbers $\mathbb{C}$, and finite fields such as $\mathbb{Z}_2$. For simplicity, in much of the discussion, we will assume real valued tensors, but substituting in values from other fields is entirely reasonable.

If scalars are rank-0 tensors, what constitutes a rank-1 tensor? Formally, speaking, a rank-1 tensor is a vector; a list of real numbers. Traditionally, vectors are written as either column vectors

$$\begin{pmatrix} a \\ b \end{pmatrix}$$

or as row vectors

$$(a \;\; b)$$

Notationally, the collection of all column vectors of length 2 is denoted $\mathbb{R}^{2\times 1}$ while the set of all row vectors of length 2 is $\mathbb{R}^{1\times 2}$. More computationally, we might say that the shape of a column vector is (2, 1), while the shape of a row vector is (1, 2). If we don't wish to specify whether a vector is a row vector or column vector, we can say it comes from the set $\mathbb{R}^2$ and has shape (2). This notion of tensor shape is quite important for understanding TensorFlow computations, and we will return to it later on in this chapter.

One of the simplest uses of vectors is to represent coordinates in the real world. Suppose that we decide on an origin point (say the position where you're currently standing). Then any position in the world can be represented by three displacement values from your current position (left-right displacement, front-back displacement, up-down displacement). Thus, the set of vectors (vector space) $\mathbb{R}^3$ can represent any position in the world.

For a different example, let's suppose that a cat is described by its height, weight, and color. Then a video game cat can be represented as a vector

$$\begin{pmatrix} \text{height} \\ \text{weight} \\ \text{color} \end{pmatrix}$$

in the space $\mathbb{R}^3$. This type of representation is often called a *featurization*. That is, a featurization is a representation of a real-world entity as a vector (or more generally as a tensor). Nearly all machine learning algorithms operate on vectors or tensors. Thus the process of featurization is a critical part of any machine learning pipeline. Often, the featurization system can be the most sophisticated part of a machine learning system. Suppose we have a benzene molecule as illustrated in Figure 2-1.

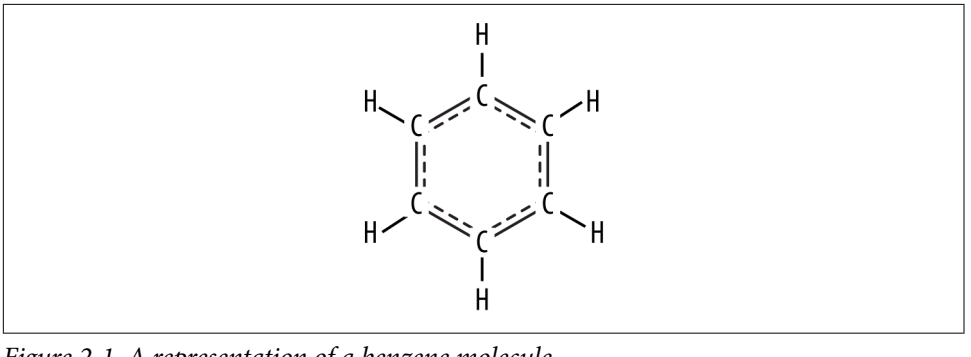

Figure 2-1. A representation of a benzene molecule.

How can we transform this molecule into a vector suitable for a query to a machine learning system? There are a number of potential solutions to this problem, most of which exploit the idea of marking the presence of subfragments of the molecule. The presence or absence of specific subfragments is marked by setting indices in a binary vector (in $\{0, 1\}^n$) to 1/0, respectively. This process is illustrated in Figure 2-2.

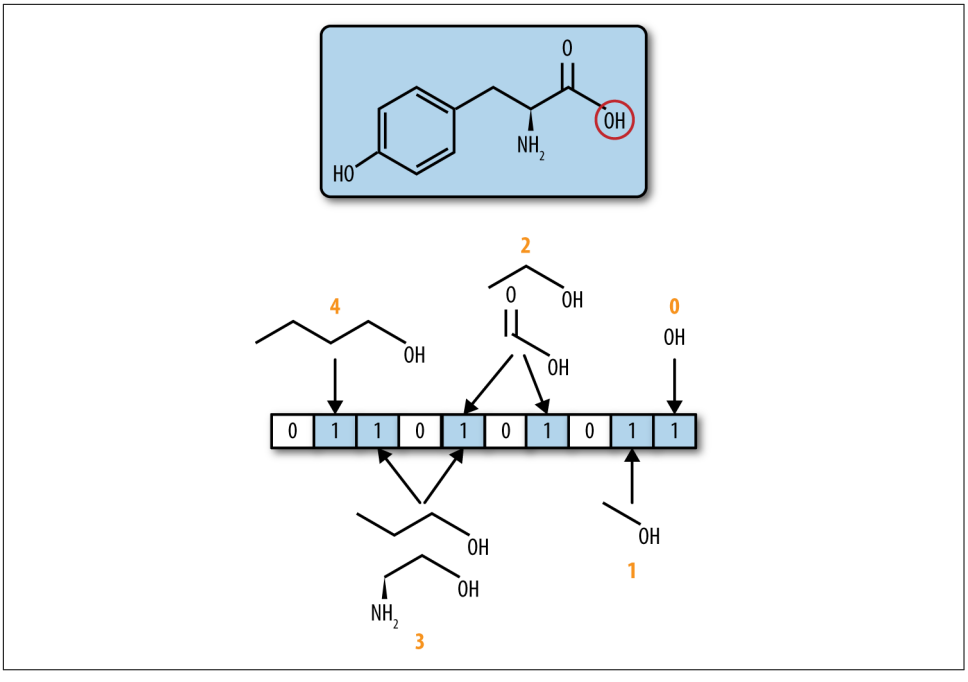

Figure 2-2. Subfragments of the molecule to be featurized are selected (those containing OH). These fragments are hashed into indices in a fixed-length vector. These positions are set to 1 and all other positions are set to 0.

Note that this process sounds (and is) fairly complex. In fact, one of the most challenging aspects of building a machine learning system is deciding how to transform the data in question into a tensorial format. For some types of data, this transformation is obvious. For others (such as molecules), the transformation required can be quite subtle. For the practitioner of machine learning, it isn't usually necessary to invent a new featurization method since the scholarly literature is extensive, but it will often be necessary to read research papers to understand best practices for transforming a new data stream.

Now that we have established that rank-0 tensors are scalars ($\mathbb{R}$) and that rank-1 tensors are vectors ($\mathbb{R}^n$), what is a rank-2 tensor? Traditionally, a rank-2 tensor is referred to as a matrix:

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

This matrix has two rows and two columns. The set of all such matrices is referred to as $\mathbb{R}^{2\times 2}$. Returning to our notion of tensor shape earlier, the shape of this matrix is

(2, 2). Matrices are traditionally used to represent transformations of vectors. For example, the action of rotating a vector in the plane by angle α can be performed by the matrix

$$R_\alpha = \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{pmatrix}$$

To see this, note that the *x* unit vector (1, 0) is transformed by matrix multiplication into the vector (cos (α), sin (α)). (We will cover the detailed definition of matrix multiplication later in the chapter, but will simply display the result for the moment).

$$\begin{pmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} \cos(\alpha) \\ \sin(\alpha) \end{pmatrix}$$

This transformation can be visualized graphically as well. Figure 2-3 demonstrates how the final vector corresponds to a rotation of the original unit vector.

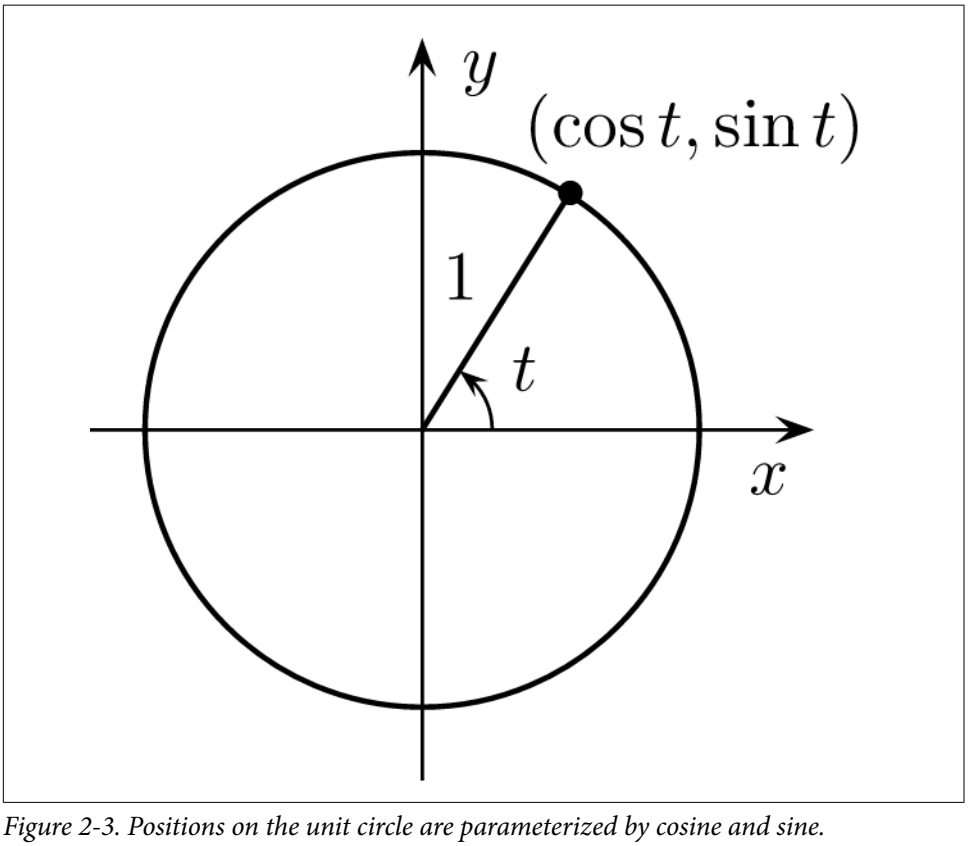

Figure 2-3. Positions on the unit circle are parameterized by cosine and sine.

Matrix Mathematics

There are a number of standard mathematical operations on matrices that machine learning programs use repeatedly. We will briefly review some of the most fundamental of these operations.

The matrix transpose is a convenient operation that flips a matrix around its diagonal. Mathematically, suppose A is a matrix; then the transpose matrix A^T is defined by equation $A^T_{ij} = A_{ji}$. For example, the transpose of the rotation matrix R_α is

$$R^T_\alpha = \begin{pmatrix} \cos(\alpha) & \sin(\alpha) \\ -\sin(\alpha) & \cos(\alpha) \end{pmatrix}$$

Addition of matrices is only defined for matrices of the same shape and is simply performed elementwise. For example:

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} + \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} 2 & 3 \\ 4 & 5 \end{pmatrix}$$

Similarly, matrices can be multiplied by scalars. In this case, each element of the matrix is simply multiplied elementwise by the scalar in question:

$$2 \cdot \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} = \begin{pmatrix} 2 & 4 \\ 6 & 8 \end{pmatrix}$$

Furthermore, it is sometimes possible to multiply two matrices directly. This notion of matrix multiplication is probably the most important mathematical concept associated with matrices. Note specifically that matrix multiplication is not the same notion as elementwise multiplication of matrices! Rather, suppose we have a matrix A of shape (m, n) with m rows and n columns. Then, A can be multiplied on the right by any matrix B of shape (n, k) (where k is any positive integer) to form matrix AB of shape (m, k). For the actual mathematical description, suppose A is a matrix of shape (m, n) and B is a matrix of shape (n, k). Then AB is defined by

$$(AB)_{ij} = \sum_k A_{ik} B_{kj}$$

We displayed a matrix multiplication equation earlier in brief. Let's expand that example now that we have the formal definition:

$$\begin{pmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} \cos(\alpha) \cdot 1 - \sin(\alpha) \cdot 0 \\ \sin(\alpha) \cdot 1 - \cos(\alpha) \cdot 0 \end{pmatrix} = \begin{pmatrix} \cos(\alpha) \\ \sin(\alpha) \end{pmatrix}$$