As another physical example, Einstein's field equations of general relativity are commonly expressed in tensorial format:

$$R_{\mu\nu} - \frac{1}{2}Rg_{\mu\nu} + \Lambda g_{\mu\nu} = \frac{8\pi G}{c^4}T_{\mu\nu}$$

Here $R_{\mu\nu}$ is the Ricci curvature tensor, $g_{\mu\nu}$ is the metric tensor, $T_{\mu\nu}$ is the stress-energy tensor, and the remaining quantities are scalars. Note, however, that there's an important subtlety distinguishing these tensors and the other tensors we've discussed previously. Quantities like the metric tensor provide a separate tensor (in the sense of an array of numbers) for each point in space-time (mathematically, the metric tensor is a tensor field). The same holds for the stress tensor previously discussed, and for the other tensors in these equations. At a given point in space-time, each of these quantities becomes a symmetric rank-2 tensor of shape (4, 4) using our notation.

Part of the power of modern tensor calculus systems such as TensorFlow is that some of the mathematical machinery long used for classical physics can now be adapted to solve applied problems in image processing and language understanding. At the same time, today's tensor calculus systems are still limited compared with the mathematical machinery of physicists. For example, there's no simple way to talk about a quantity such as the metric tensor using TensorFlow yet. We hope that as tensor calculus becomes more fundamental to computer science, the situation will change and that systems like TensorFlow will serve as a bridge between the physical world and the computational world.

## Mathematical Asides

The discussion so far in this chapter has introduced tensors informally via example and illustration. In our definition, a tensor is simply an array of numbers. It's often convenient to view a tensor as a function instead. The most common definition introduces a tensor as a multilinear function from a product of vector spaces to the real numbers:

$$T: V_1 \times V_2 \times \cdots V_n \to \mathbb{R}$$

This definition uses a number of terms you haven't seen. A vector space is simply a collection of vectors. You've seen a few examples of vector spaces such as $\mathbb{R}^3$ or generally $\mathbb{R}^n$. We won't lose any generality by holding that $V_i = \mathbb{R}^{d_i}$. As we defined previously, a function $f$ is linear if $f(x + y) = f(x) + f(y)$ and $f(cx) = cf(x)$. A multilinear function is simply a function that is linear in each argument. This function can be

viewed as assigning individual entries of a multidimensional array, when provided indices into the array as arguments.

We won't use this more mathematical definition much in this book, but it serves as a useful bridge to connect the deep learning concepts you will learn about with the centuries of mathematical research that have been undertaken on tensors by the physics and mathematics communities.

#### Covariance and Contravariance

Our definition here has swept many details under the rug that would need to be carefully attended to for a formal treatment. For example, we don't touch upon the notion of covariant and contravariant indices here. What we call a rank-$n$ tensor is better described as a $(p, q)$-tensor where $n = p + q$ and $p$ is the number of contravariant indices, and $q$ the number of covariant indices. Matrices are (1,1)-tensors, for example. As a subtlety, there are rank-2 tensors that are not matrices! We won't dig into these topics carefully here since they don't crop up much in machine learning, but we encourage you to understand how covariance and contravariance affect the machine learning systems you construct.

# Basic Computations in TensorFlow

We've spent the last sections covering the mathematical definitions of various tensors. It's now time to cover how to create and manipulate tensors using TensorFlow. For this section, we recommend you follow along using an interactive Python session (with IPython). Many of the basic TensorFlow concepts are easiest to understand after experimenting with them directly.

## Installing TensorFlow and Getting Started

Before continuing this section, you will need to install TensorFlow on your machine. The details of installation will vary depending on your particular hardware, so we refer you to the official TensorFlow documentation for more details.

Although there are frontends to TensorFlow in multiple programming languages, we will exclusively use the TensorFlow Python API in the remainder of this book. We recommend that you install Anaconda Python, which packages many useful numerical libraries along with the base Python executable.

Once you've installed TensorFlow, we recommend that you invoke it interactively while you're learning the basic API (see Example 2-1). When experimenting with TensorFlow interactively, it's convenient to use `tf.InteractiveSession()`. Invoking this statement within IPython (an interactive Python shell) will make TensorFlow