CHAPTER 4

Fully Connected Deep Networks

This chapter will introduce you to fully connected deep networks. Fully connected networks are the workhorses of deep learning, used for thousands of applications. The major advantage of fully connected networks is that they are “structure agnostic.” That is, no special assumptions need to be made about the input (for example, that the input consists of images or videos). We will make use of this generality to use fully connected deep networks to address a problem in chemical modeling later in this chapter.

We delve briefly into the mathematical theory underpinning fully connected networks. In particular, we explore the concept that fully connected architectures are “universal approximators” capable of learning any function. This concept provides an explanation of the generality of fully connected architectures, but comes with many caveats that we discuss at some depth.

While being structure agnostic makes fully connected networks very broadly applicable, such networks do tend to have weaker performance than special-purpose networks tuned to the structure of a problem space. We will discuss some of the limitations of fully connected architectures later in this chapter.

What Is a Fully Connected Deep Network?

A fully connected neural network consists of a series of fully connected layers. A fully connected layer is a function from $\mathbb{R}^m$ to $\mathbb{R}^n$. Each output dimension depends on each input dimension. Pictorially, a fully connected layer is represented as follows in Figure 4-1.

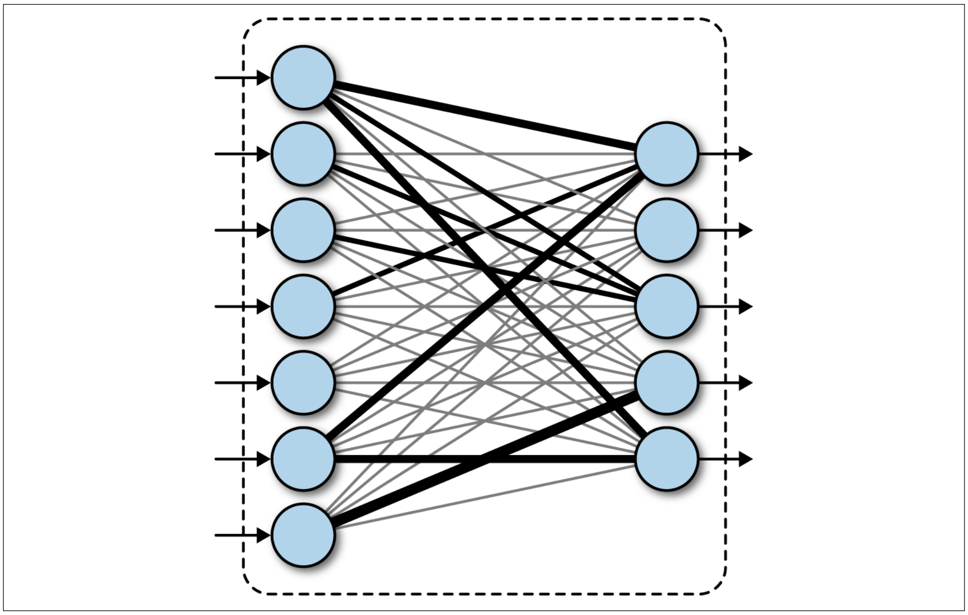

Figure 4-1. A fully connected layer in a deep network.

Let's dig a little deeper into what the mathematical form of a fully connected network is. Let $x \in \mathbb{R}^m$ represent the input to a fully connected layer. Let $y_i \in \mathbb{R}$ be the i-th output from the fully connected layer. Then $y_i \in \mathbb{R}$ is computed as follows:

$$y_i = \sigma\left(w_1 x_1 + \cdots + w_m x_m\right)$$

Here, σ is a nonlinear function (for now, think of σ as the sigmoid function introduced in the previous chapter), and the w_i are learnable parameters in the network. The full output y is then

$$y = \begin{pmatrix} \sigma\left(w_{1,1} x_1 + \cdots + w_{1,m} x_m\right) \\ \vdots \\ \sigma\left(w_{n,1} x_1 + \cdots + w_{n,m} x_m\right) \end{pmatrix}$$

Note that it's directly possible to stack fully connected networks. A network with multiple fully connected networks is often called a "deep" network as depicted in Figure 4-2.

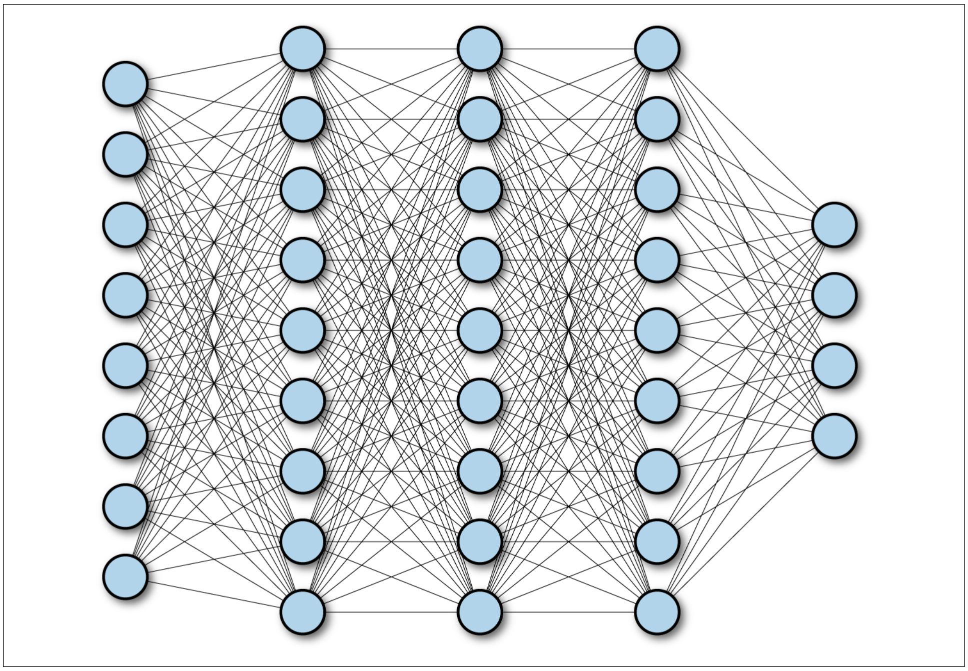

Figure 4-2. A multilayer deep fully connected network.

As a quick implementation note, note that the equation for a single neuron looks very similar to a dot-product of two vectors (recall the discussion of tensor basics). For a layer of neurons, it is often convenient for efficiency purposes to compute y as a matrix multiply:

$$y = \sigma(wx)$$

where sigma is a matrix in $\mathbb{R}^{n \times m}$ and the nonlinearity σ is applied componentwise.

"Neurons" in Fully Connected Networks

The nodes in fully connected networks are commonly referred to as "neurons." Consequently, elsewhere in the literature, fully connected networks will commonly be referred to as "neural networks." This nomenclature is largely a historical accident.

In the 1940s, Warren S. McCulloch and Walter Pitts published a first mathematical model of the brain that argued that neurons were capable of computing arbitrary functions on Boolean quantities. Successors to this work slightly refined this logical model by making mathematical "neurons" continuous functions that varied between zero and one. If the inputs of these functions grew large enough, the neuron "fired"