length of the chain gives the **depth** of the model. It is from this terminology that the name "deep learning" arises. The final layer of a feedforward network is called the **output layer**. During neural network training, we drive $f(\boldsymbol{x})$ to match $f^*(\boldsymbol{x})$. The training data provides us with noisy, approximate examples of $f^*(\boldsymbol{x})$ evaluated at different training points. Each example $\boldsymbol{x}$ is accompanied by a label $y \approx f^*(\boldsymbol{x})$. The training examples specify directly what the output layer must do at each point $\boldsymbol{x}$; it must produce a value that is close to y. The behavior of the other layers is not directly specified by the training data. The learning algorithm must decide how to use those layers to produce the desired output, but the training data does not say what each individual layer should do. Instead, the learning algorithm must decide how to use these layers to best implement an approximation of f^*. Because the training data does not show the desired output for each of these layers, these layers are called **hidden layers**.

Finally, these networks are called *neural* because they are loosely inspired by neuroscience. Each hidden layer of the network is typically vector-valued. The dimensionality of these hidden layers determines the **width** of the model. Each element of the vector may be interpreted as playing a role analogous to a neuron. Rather than thinking of the layer as representing a single vector-to-vector function, we can also think of the layer as consisting of many **units** that act in parallel, each representing a vector-to-scalar function. Each unit resembles a neuron in the sense that it receives input from many other units and computes its own activation value. The idea of using many layers of vector-valued representation is drawn from neuroscience. The choice of the functions $f^{(i)}(\boldsymbol{x})$ used to compute these representations is also loosely guided by neuroscientific observations about the functions that biological neurons compute. However, modern neural network research is guided by many mathematical and engineering disciplines, and the goal of neural networks is not to perfectly model the brain. It is best to think of feedforward networks as function approximation machines that are designed to achieve statistical generalization, occasionally drawing some insights from what we know about the brain, rather than as models of brain function.

One way to understand feedforward networks is to begin with linear models and consider how to overcome their limitations. Linear models, such as logistic regression and linear regression, are appealing because they may be fit efficiently and reliably, either in closed form or with convex optimization. Linear models also have the obvious defect that the model capacity is limited to linear functions, so the model cannot understand the interaction between any two input variables.

To extend linear models to represent nonlinear functions of $\boldsymbol{x}$, we can apply the linear model not to $\boldsymbol{x}$ itself but to a transformed input $\phi(\boldsymbol{x})$, where ϕ is a