$$a^l_j = \sigma\left(\sum_k w^l_{jk} a^{l-1}_k + b^l_j\right), \tag{2.1}$$

where the sum is over all neurons k in the $(l-1)$-th layer. To rewrite this expression in a matrix form we define a *weight matrix* w^l for each layer, l. The entries of the weight matrix w^l are just the weights connecting to the l-th layer of neurons, that is, the entry in the j-th row and k-th column is w^l_{jk}. Similarly, for each layer l we define a *bias vector*, b^l. You can probably guess how this works – the components of the bias vector are just the values b^l_j, one component for each neuron in the l-th layer. And finally, we define an activation vector al whose components are the activations a^l_j. The last ingredient we need to rewrite 2.1 in a matrix form is the idea of vectorizing a function such as σ. We met vectorization briefly in the last chapter, but to recap, the idea is that we want to apply a function such as σ to every element in a vector v. We use the obvious notation $\sigma(v)$ to denote this kind of elementwise application of a function. That is, the components of $\sigma(v)$ are just $\sigma(v)_j = \sigma(v_j)$. As an example, if we have the function $f(x) = x^2$ then the vectorized form of f has the effect

$$f\left(\begin{bmatrix} 2 \\ 3 \end{bmatrix}\right) = \begin{bmatrix} f(2) \\ f(3) \end{bmatrix} = \begin{bmatrix} 4 \\ 9 \end{bmatrix}, \tag{2.2}$$

that is, the vectorized f just squares every element of the vector.

With these notations in mind, Equation 2.1 can be rewritten in the beautiful and compact vectorized form

$$a^l = \sigma(w^l a^{l-1} + b^l). \tag{2.3}$$

This expression gives us a much more global way of thinking about how the activations in one layer relate to activations in the previous layer: we just apply the weight matrix to the activations, then add the bias vector, and finally apply the σ function[1]. That global view is often easier and more succinct (and involves fewer indices!) than the neuron-by-neuron view we've taken to now. Think of it as a way of escaping index hell, while remaining precise about what's going on. The expression is also useful in practice, because most matrix libraries provide fast ways of implementing matrix multiplication, vector addition, and vectorization. Indeed, the code (see 1.6) in the last chapter made implicit use of this expression to compute the behaviour of the network.

When using Equation 2.3 to compute a^l, we compute the intermediate quantity $z^l \equiv w^l a^{l-1} + b^l$ along the way. This quantity turns out to be useful enough to be worth naming: we call z^l the weighted input to the neurons in layer l. We'll make considerable use of the weighted input z^l later in the chapter. Equation 2.3 is sometimes written in terms of the weighted input, as $a^l = \sigma(z^l)$. It's also worth noting that z^l has components $z^l_j = \sum_k w^l_{jk} a^{l-1}_k + b^l_j$, that is, z^l_j is just the weighted input to the activation function for neuron j in layer l.

[1] By the way, it's this expression that motivates the quirk in the w^l_{jk} notation mentioned earlier. If we used j to index the input neuron, and k to index the output neuron, then we'd need to replace the weight matrix in Equation 2.3 by the transpose of the weight matrix. That's a small change, but annoying, and we'd lose the easy simplicity of saying (and thinking) "apply the weight matrix to the activations".