feeding into a particular hidden unit, then, for a given input data point, the sign of the pre-activation of the hidden unit will be reversed, and therefore so too will the activation, because tanh is an odd function, so that $\tanh(-a) = -\tanh(a)$. This transformation can be exactly compensated for by changing the sign of all the weights leading out of that hidden unit. Thus, by changing the signs of a particular group of weights (and a bias), the input–output mapping function represented by the network is unchanged, and so we have found two different weight vectors that give rise to the same mapping function. For M hidden units, there will be M such 'sign-flip' symmetries, and thus, any given weight vector will be one of a set 2^M equivalent weight vectors .

Similarly, imagine that we interchange the values of all of the weights (and the bias) leading both into and out of a particular hidden unit with the corresponding values of the weights (and bias) associated with a different hidden unit. Again, this clearly leaves the network input–output mapping function unchanged, but it corresponds to a different choice of weight vector. For M hidden units, any given weight vector will belong to a set of $M \times (M-1) \times \cdots \times 2 \times 1 = M!$ equivalent weight vectors associated with this interchange symmetry, corresponding to the $M!$ different orderings of the hidden units. The network will therefore have an overall weight-space symmetry factor of $M!\,2^M$. For networks with more than two layers of weights, the total level of symmetry will be given by the product of such factors, one for each layer of hidden units.

It turns out that these factors account for all the symmetries in weight space (except for possible accidental symmetries due to specific choices for the weight values). Furthermore, the existence of these symmetries is not a particular property of the tanh function but applies to a wide range of activation functions (Kurková and Kainen, 1994). In general, these symmetries in weight space are of little practical consequence, since network training aims to find a specific setting for the parameters, and the existence of other, equivalent, settings is of little consequence. However, weight-space symmetries do play a role when Bayesian methods are used to evaluate the probability distribution over networks of different sizes (Bishop, 2006).

6.3. Deep Networks

We have motivated the development of neural networks by making the basis functions of a linear regression or classification model themselves be governed by learnable parameters, giving rise to the two-layer network model shown in Figure 6.9. For many years, this was the most widely used architecture, primarily because it proved difficult to train networks with more than two layers effectively. However, extending neural networks to have more than two layers, known as deep neural networks, brings many advantages as we will discuss shortly, and recent advances in techniques for training neural networks are effective for networks with many layers.

Chapter 7

We can easily extend the two-layer network architecture (6.12) to any finite number L of layers, in which layer $l = 1, \ldots, L$ computes the following function:

$$\mathbf{z}^{(l)} = h^{(l)}\left(\mathbf{W}^{(l)}\mathbf{z}^{(l-1)}\right) \tag{6.19}$$