6.3.3 Other Hidden Units

Many other types of hidden units are possible, but are used less frequently.

In general, a wide variety of differentiable functions perform perfectly well. Many unpublished activation functions perform just as well as the popular ones. To provide a concrete example, the authors tested a feedforward network using $\boldsymbol{h} = \cos(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b})$ on the MNIST dataset and obtained an error rate of less than 1%, which is competitive with results obtained using more conventional activation functions. During research and development of new techniques, it is common to test many different activation functions and find that several variations on standard practice perform comparably. This means that usually new hidden unit types are published only if they are clearly demonstrated to provide a significant improvement. New hidden unit types that perform roughly comparably to known types are so common as to be uninteresting.

It would be impractical to list all of the hidden unit types that have appeared in the literature. We highlight a few especially useful and distinctive ones.

One possibility is to not have an activation $g(z)$ at all. One can also think of this as using the identity function as the activation function. We have already seen that a linear unit can be useful as the output of a neural network. It may also be used as a hidden unit. If every layer of the neural network consists of only linear transformations, then the network as a whole will be linear. However, it is acceptable for some layers of the neural network to be purely linear. Consider a neural network layer with n inputs and p outputs, $\boldsymbol{h} = g(\boldsymbol{W}^\top \boldsymbol{x} + \boldsymbol{b})$. We may replace this with two layers, with one layer using weight matrix $\boldsymbol{U}$ and the other using weight matrix $\boldsymbol{V}$. If the first layer has no activation function, then we have essentially factored the weight matrix of the original layer based on $\boldsymbol{W}$. The factored approach is to compute $\boldsymbol{h} = g(\boldsymbol{V}^\top \boldsymbol{U}^\top \boldsymbol{x} + \boldsymbol{b})$. If $\boldsymbol{U}$ produces q outputs, then $\boldsymbol{U}$ and $\boldsymbol{V}$ together contain only $(n + p)q$ parameters, while $\boldsymbol{W}$ contains np parameters. For small q, this can be a considerable saving in parameters. It comes at the cost of constraining the linear transformation to be low-rank, but these low-rank relationships are often sufficient. Linear hidden units thus offer an effective way of reducing the number of parameters in a network.

Softmax units are another kind of unit that is usually used as an output (as described in section 6.2.2.3) but may sometimes be used as a hidden unit. Softmax units naturally represent a probability distribution over a discrete variable with k possible values, so they may be used as a kind of switch. These kinds of hidden units are usually only used in more advanced architectures that explicitly learn to manipulate memory, described in section 10.12.