increasing its weights, it is possible to eventually obtain a network with reasonable initial activations throughout. If learning is still too slow at this point, it can be useful to look at the range or standard deviation of the gradients as well as the activations. This procedure can in principle be automated and is generally less computationally costly than hyperparameter optimization based on validation set error because it is based on feedback from the behavior of the initial model on a single batch of data, rather than on feedback from a trained model on the validation set. While long used heuristically, this protocol has recently been specified more formally and studied by Mishkin and Matas (2015).

So far we have focused on the initialization of the weights. Fortunately, initialization of other parameters is typically easier.

The approach for setting the biases must be coordinated with the approach for settings the weights. Setting the biases to zero is compatible with most weight initialization schemes. There are a few situations where we may set some biases to non-zero values:

- If a bias is for an output unit, then it is often beneficial to initialize the bias to obtain the right marginal statistics of the output. To do this, we assume that the initial weights are small enough that the output of the unit is determined only by the bias. This justifies setting the bias to the inverse of the activation function applied to the marginal statistics of the output in the training set. For example, if the output is a distribution over classes and this distribution is a highly skewed distribution with the marginal probability of class i given by element c_i of some vector $\boldsymbol{c}$, then we can set the bias vector $\boldsymbol{b}$ by solving the equation $\text{softmax}(\boldsymbol{b}) = \boldsymbol{c}$. This applies not only to classifiers but also to models we will encounter in Part III, such as autoencoders and Boltzmann machines. These models have layers whose output should resemble the input data $\boldsymbol{x}$, and it can be very helpful to initialize the biases of such layers to match the marginal distribution over $\boldsymbol{x}$.

- Sometimes we may want to choose the bias to avoid causing too much saturation at initialization. For example, we may set the bias of a ReLU hidden unit to 0.1 rather than 0 to avoid saturating the ReLU at initialization. This approach is not compatible with weight initialization schemes that do not expect strong input from the biases though. For example, it is not recommended for use with random walk initialization (Sussillo, 2014).

- Sometimes a unit controls whether other units are able to participate in a function. In such situations, we have a unit with output u and another unit $h \in [0, 1]$, and they are multiplied together to produce an output uh. We