also possible. To make predictions we must re-normalize the ensemble:

$$p_{\text{ensemble}}(y \mid \boldsymbol{x}) = \frac{\tilde{p}_{\text{ensemble}}(y \mid \boldsymbol{x})}{\sum_{y'} \tilde{p}_{\text{ensemble}}(y' \mid \boldsymbol{x})}. \tag{7.55}$$

A key insight (Hinton *et al.*, 2012c) involved in dropout is that we can approximate p_{ensemble} by evaluating $p(y \mid \boldsymbol{x})$ in one model: the model with all units, but with the weights going out of unit i multiplied by the probability of including unit i. The motivation for this modification is to capture the right expected value of the output from that unit. We call this approach the **weight scaling inference rule**. There is not yet any theoretical argument for the accuracy of this approximate inference rule in deep nonlinear networks, but empirically it performs very well.

Because we usually use an inclusion probability of $\frac{1}{2}$, the weight scaling rule usually amounts to dividing the weights by 2 at the end of training, and then using the model as usual. Another way to achieve the same result is to multiply the states of the units by 2 during training. Either way, the goal is to make sure that the expected total input to a unit at test time is roughly the same as the expected total input to that unit at train time, even though half the units at train time are missing on average.

For many classes of models that do not have nonlinear hidden units, the weight scaling inference rule is exact. For a simple example, consider a softmax regression classifier with n input variables represented by the vector $\mathbf{v}$:

$$P(\mathrm{y} = y \mid \mathbf{v}) = \text{softmax}\left(\boldsymbol{W}^\top \mathbf{v} + \boldsymbol{b}\right)_y. \tag{7.56}$$

We can index into the family of sub-models by element-wise multiplication of the input with a binary vector d:

$$P(\mathrm{y} = y \mid \mathbf{v}; \boldsymbol{d}) = \text{softmax}\left(\boldsymbol{W}^\top (\boldsymbol{d} \odot \mathbf{v}) + \boldsymbol{b}\right)_y. \tag{7.57}$$

The ensemble predictor is defined by re-normalizing the geometric mean over all ensemble members' predictions:

$$P_{\text{ensemble}}(\mathrm{y} = y \mid \mathbf{v}) = \frac{\tilde{P}_{\text{ensemble}}(\mathrm{y} = y \mid \mathbf{v})}{\sum_{y'} \tilde{P}_{\text{ensemble}}(\mathrm{y} = y' \mid \mathbf{v})} \tag{7.58}$$

where

$$\tilde{P}_{\text{ensemble}}(\mathrm{y} = y \mid \mathbf{v}) = \sqrt[2^n]{\prod_{\boldsymbol{d} \in \{0,1\}^n} P(\mathrm{y} = y \mid \mathbf{v}; \boldsymbol{d})}. \tag{7.59}$$