

the fraction of counts of each outcome observed in the training set:

$$\text{softmax}(\mathbf{z}(\mathbf{x}; \boldsymbol{\theta}))_i \approx \frac{\sum_{j=1}^m \mathbf{1}_{y^{(j)}=i, \mathbf{x}^{(j)}=\mathbf{x}}}{\sum_{j=1}^m \mathbf{1}_{\mathbf{x}^{(j)}=\mathbf{x}}}. \quad (6.31)$$

Because maximum likelihood is a consistent estimator, this is guaranteed to happen so long as the model family is capable of representing the training distribution. In practice, limited model capacity and imperfect optimization will mean that the model is only able to approximate these fractions.

Many objective functions other than the log-likelihood do not work as well with the softmax function. Specifically, objective functions that do not use a log to undo the exp of the softmax fail to learn when the argument to the exp becomes very negative, causing the gradient to vanish. In particular, squared error is a poor loss function for softmax units, and can fail to train the model to change its output, even when the model makes highly confident incorrect predictions (Bridle, 1990). To understand why these other loss functions can fail, we need to examine the softmax function itself.

Like the sigmoid, the softmax activation can saturate. The sigmoid function has a single output that saturates when its input is extremely negative or extremely positive. In the case of the softmax, there are multiple output values. These output values can saturate when the differences between input values become extreme. When the softmax saturates, many cost functions based on the softmax also saturate, unless they are able to invert the saturating activating function.

To see that the softmax function responds to the difference between its inputs, observe that the softmax output is invariant to adding the same scalar to all of its inputs:

$$\text{softmax}(\mathbf{z}) = \text{softmax}(\mathbf{z} + c). \quad (6.32)$$

Using this property, we can derive a numerically stable variant of the softmax:

$$\text{softmax}(\mathbf{z}) = \text{softmax}(\mathbf{z} - \max_i z_i). \quad (6.33)$$

The reformulated version allows us to evaluate softmax with only small numerical errors even when \mathbf{z} contains extremely large or extremely negative numbers. Examining the numerically stable variant, we see that the softmax function is driven by the amount that its arguments deviate from $\max_i z_i$.

An output $\text{softmax}(\mathbf{z})_i$ saturates to 1 when the corresponding input is maximal ($z_i = \max_i z_i$) and z_i is much greater than all of the other inputs. The output $\text{softmax}(\mathbf{z})_i$ can also saturate to 0 when z_i is not maximal and the maximum is much greater. This is a generalization of the way that sigmoid units saturate, and