To generalize to the case of a discrete variable with n values, we now need to produce a vector $\hat{\boldsymbol{y}}$, with $\hat{y}_i = P(y = i \mid \boldsymbol{x})$. We require not only that each element of $\hat{y}_i$ be between 0 and 1, but also that the entire vector sums to 1 so that it represents a valid probability distribution. The same approach that worked for the Bernoulli distribution generalizes to the multinoulli distribution. First, a linear layer predicts unnormalized log probabilities:

$$\boldsymbol{z} = \boldsymbol{W}^\top \boldsymbol{h} + \boldsymbol{b}, \tag{6.28}$$

where $z_i = \log \tilde{P}(y = i \mid \boldsymbol{x})$. The softmax function can then exponentiate and normalize $\boldsymbol{z}$ to obtain the desired $\hat{\boldsymbol{y}}$. Formally, the softmax function is given by

$$\text{softmax}(\boldsymbol{z})_i = \frac{\exp(z_i)}{\sum_j \exp(z_j)}. \tag{6.29}$$

As with the logistic sigmoid, the use of the exp function works very well when training the softmax to output a target value y using maximum log-likelihood. In this case, we wish to maximize $\log P(\mathrm{y} = i; \boldsymbol{z}) = \log \text{softmax}(\boldsymbol{z})_i$. Defining the softmax in terms of exp is natural because the log in the log-likelihood can undo the exp of the softmax:

$$\log \text{softmax}(\boldsymbol{z})_i = z_i - \log \sum_j \exp(z_j). \tag{6.30}$$

The first term of equation 6.30 shows that the input z_i always has a direct contribution to the cost function. Because this term cannot saturate, we know that learning can proceed, even if the contribution of z_i to the second term of equation 6.30 becomes very small. When maximizing the log-likelihood, the first term encourages z_i to be pushed up, while the second term encourages all of $\boldsymbol{z}$ to be pushed down. To gain some intuition for the second term, $\log \sum_j \exp(z_j)$, observe that this term can be roughly approximated by $\max_j z_j$. This approximation is based on the idea that $\exp(z_k)$ is insignificant for any z_k that is noticeably less than $\max_j z_j$. The intuition we can gain from this approximation is that the negative log-likelihood cost function always strongly penalizes the most active incorrect prediction. If the correct answer already has the largest input to the softmax, then the $-z_i$ term and the $\log \sum_j \exp(z_j) \approx \max_j z_j = z_i$ terms will roughly cancel. This example will then contribute little to the overall training cost, which will be dominated by other examples that are not yet correctly classified.

So far we have discussed only a single example. Overall, unregularized maximum likelihood will drive the model to learn parameters that drive the softmax to predict