

as the cross-entropy between the training data and the model distribution. This cost function is given by

$$J(\boldsymbol{\theta}) = -\mathbb{E}_{\mathbf{x}, \mathbf{y} \sim \hat{p}_{\text{data}}} \log p_{\text{model}}(\mathbf{y} \mid \mathbf{x}). \quad (6.12)$$

The specific form of the cost function changes from model to model, depending on the specific form of $\log p_{\text{model}}$. The expansion of the above equation typically yields some terms that do not depend on the model parameters and may be discarded. For example, as we saw in section 5.5.1, if $p_{\text{model}}(\mathbf{y} \mid \mathbf{x}) = \mathcal{N}(\mathbf{y}; f(\mathbf{x}; \boldsymbol{\theta}), \mathbf{I})$, then we recover the mean squared error cost,

$$J(\theta) = \frac{1}{2} \mathbb{E}_{\mathbf{x}, \mathbf{y} \sim \hat{p}_{\text{data}}} \|\mathbf{y} - f(\mathbf{x}; \boldsymbol{\theta})\|^2 + \text{const}, \quad (6.13)$$

up to a scaling factor of $\frac{1}{2}$ and a term that does not depend on $\boldsymbol{\theta}$. The discarded constant is based on the variance of the Gaussian distribution, which in this case we chose not to parametrize. Previously, we saw that the equivalence between maximum likelihood estimation with an output distribution and minimization of mean squared error holds for a linear model, but in fact, the equivalence holds regardless of the $f(\mathbf{x}; \boldsymbol{\theta})$ used to predict the mean of the Gaussian.

An advantage of this approach of deriving the cost function from maximum likelihood is that it removes the burden of designing cost functions for each model. Specifying a model $p(\mathbf{y} \mid \mathbf{x})$ automatically determines a cost function $\log p(\mathbf{y} \mid \mathbf{x})$.

One recurring theme throughout neural network design is that the gradient of the cost function must be large and predictable enough to serve as a good guide for the learning algorithm. Functions that saturate (become very flat) undermine this objective because they make the gradient become very small. In many cases this happens because the activation functions used to produce the output of the hidden units or the output units saturate. The negative log-likelihood helps to avoid this problem for many models. Many output units involve an exp function that can saturate when its argument is very negative. The log function in the negative log-likelihood cost function undoes the exp of some output units. We will discuss the interaction between the cost function and the choice of output unit in section 6.2.2.

One unusual property of the cross-entropy cost used to perform maximum likelihood estimation is that it usually does not have a minimum value when applied to the models commonly used in practice. For discrete output variables, most models are parametrized in such a way that they cannot represent a probability of zero or one, but can come arbitrarily close to doing so. Logistic regression is an example of such a model. For real-valued output variables, if the model