a good cost function for nearly any kind of output layer.

In general, if we define a conditional distribution $p(\boldsymbol{y} \mid \boldsymbol{x}; \boldsymbol{\theta})$, the principle of maximum likelihood suggests we use $-\log p(\boldsymbol{y} \mid \boldsymbol{x}; \boldsymbol{\theta})$ as our cost function.

In general, we can think of the neural network as representing a function $f(\boldsymbol{x};\boldsymbol{\theta})$. The outputs of this function are not direct predictions of the value $\boldsymbol{y}$. Instead, $f(\boldsymbol{x};\boldsymbol{\theta}) = \boldsymbol{\omega}$ provides the parameters for a distribution over y. Our loss function can then be interpreted as $-\log p(\mathbf{y}; \boldsymbol{\omega}(\boldsymbol{x}))$.

For example, we may wish to learn the variance of a conditional Gaussian for $\mathbf{y}$, given $\mathbf{x}$. In the simple case, where the variance σ^2 is a constant, there is a closed form expression because the maximum likelihood estimator of variance is simply the empirical mean of the squared difference between observations $\mathbf{y}$ and their expected value. A computationally more expensive approach that does not require writing special-case code is to simply include the variance as one of the properties of the distribution $p(\mathbf{y} \mid \boldsymbol{x})$ that is controlled by $\boldsymbol{\omega} = f(\boldsymbol{x}; \boldsymbol{\theta})$. The negative log-likelihood $-\log p(\boldsymbol{y}; \boldsymbol{\omega}(\boldsymbol{x}))$ will then provide a cost function with the appropriate terms necessary to make our optimization procedure incrementally learn the variance. In the simple case where the standard deviation does not depend on the input, we can make a new parameter in the network that is copied directly into $\boldsymbol{\omega}$. This new parameter might be σ itself or could be a parameter v representing σ^2 or it could be a parameter β representing $\frac{1}{\sigma^2}$, depending on how we choose to parametrize the distribution. We may wish our model to predict a different amount of variance in $\mathbf{y}$ for different values of $\mathbf{x}$. This is called a **heteroscedastic** model. In the heteroscedastic case, we simply make the specification of the variance be one of the values output by $f(\mathbf{x};\boldsymbol{\theta})$. A typical way to do this is to formulate the Gaussian distribution using precision, rather than variance, as described in equation 3.22. In the multivariate case it is most common to use a diagonal precision matrix

$$\text{diag}(\boldsymbol{\beta}). \tag{6.34}$$

This formulation works well with gradient descent because the formula for the log-likelihood of the Gaussian distribution parametrized by $\boldsymbol{\beta}$ involves only multiplication by β_i and addition of $\log \boldsymbol{\beta}_i$. The gradient of multiplication, addition, and logarithm operations is well-behaved. By comparison, if we parametrized the output in terms of variance, we would need to use division. The division function becomes arbitrarily steep near zero. While large gradients can help learning, arbitrarily large gradients usually result in instability. If we parametrized the output in terms of standard deviation, the log-likelihood would still involve division, and would also involve squaring. The gradient through the squaring operation can vanish near zero, making it difficult to learn parameters that are squared.