error in $\boldsymbol{v}$ given the code of the other units. We can thus think of sparse coding as an iterative autoencoder, that repeatedly encodes and decodes its input, attempting to fix mistakes in the reconstruction after each iteration.

In this example, we have derived an update rule that updates a single unit at a time. It would be advantageous to be able to update more units simultaneously. Some graphical models, such as deep Boltzmann machines, are structured in such a way that we can solve for many entries of $\hat{\boldsymbol{h}}$ simultaneously. Unfortunately, binary sparse coding does not admit such block updates. Instead, we can use a heuristic technique called **damping** to perform block updates. In the damping approach, we solve for the individually optimal values of every element of $\hat{\boldsymbol{h}}$, then move all of the values in a small step in that direction. This approach is no longer guaranteed to increase $\mathcal{L}$ at each step, but works well in practice for many models. See Koller and Friedman (2009) for more information about choosing the degree of synchrony and damping strategies in message passing algorithms.

19.4.2 Calculus of Variations

Before continuing with our presentation of variational learning, we must briefly introduce an important set of mathematical tools used in variational learning: **calculus of variations**.

Many machine learning techniques are based on minimizing a function $J(\boldsymbol{\theta})$ by finding the input vector $\boldsymbol{\theta} \in \mathbb{R}^n$ for which it takes on its minimal value. This can be accomplished with multivariate calculus and linear algebra, by solving for the critical points where $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbf{0}$. In some cases, we actually want to solve for a function $f(\boldsymbol{x})$, such as when we want to find the probability density function over some random variable. This is what calculus of variations enables us to do.

A function of a function f is known as a **functional** $J[f]$. Much as we can take partial derivatives of a function with respect to elements of its vector-valued argument, we can take **functional derivatives**, also known as **variational derivatives**, of a functional $J[f]$ with respect to individual values of the function $f(\boldsymbol{x})$ at any specific value of $\boldsymbol{x}$. The functional derivative of the functional J with respect to the value of the function f at point $\boldsymbol{x}$ is denoted $\frac{\delta}{\delta f(x)} J$.

A complete formal development of functional derivatives is beyond the scope of this book. For our purposes, it is sufficient to state that for differentiable functions $f(\boldsymbol{x})$ and differentiable functions $g(y, \boldsymbol{x})$ with continuous derivatives, that

$$\frac{\delta}{\delta f(\boldsymbol{x})} \int g\left(f(\boldsymbol{x}), \boldsymbol{x}\right) d\boldsymbol{x} = \frac{\partial}{\partial y} g(f(\boldsymbol{x}), \boldsymbol{x}). \tag{19.46}$$