time signals to learn invariant features (Wiskott and Sejnowski, 2002).

Slow feature analysis is motivated by a general principle called the slowness principle. The idea is that the important characteristics of scenes change very slowly compared to the individual measurements that make up a description of a scene. For example, in computer vision, individual pixel values can change very rapidly. If a zebra moves from left to right across the image, an individual pixel will rapidly change from black to white and back again as the zebra's stripes pass over the pixel. By comparison, the feature indicating whether a zebra is in the image will not change at all, and the feature describing the zebra's position will change slowly. We therefore may wish to regularize our model to learn features that change slowly over time.

The slowness principle predates slow feature analysis and has been applied to a wide variety of models (Hinton, 1989; Földiák, 1989; Mobahi *et al.*, 2009; Bergstra and Bengio, 2009). In general, we can apply the slowness principle to any differentiable model trained with gradient descent. The slowness principle may be introduced by adding a term to the cost function of the form

$$\lambda \sum_t L(f(\boldsymbol{x}^{(t+1)}), f(\boldsymbol{x}^{(t)})) \tag{13.7}$$

where λ is a hyperparameter determining the strength of the slowness regularization term, t is the index into a time sequence of examples, f is the feature extractor to be regularized, and L is a loss function measuring the distance between $f(\boldsymbol{x}^{(t)})$ and $f(\boldsymbol{x}^{(t+1)})$. A common choice for L is the mean squared difference.

Slow feature analysis is a particularly efficient application of the slowness principle. It is efficient because it is applied to a linear feature extractor, and can thus be trained in closed form. Like some variants of ICA, SFA is not quite a generative model per se, in the sense that it defines a linear map between input space and feature space but does not define a prior over feature space and thus does not impose a distribution $p(\boldsymbol{x})$ on input space.

The SFA algorithm (Wiskott and Sejnowski, 2002) consists of defining $f(\boldsymbol{x}; \boldsymbol{\theta})$ to be a linear transformation, and solving the optimization problem

$$\min_{\boldsymbol{\theta}} \mathbb{E}_t (f(\boldsymbol{x}^{(t+1)})_i - f(\boldsymbol{x}^{(t)})_i)^2 \tag{13.8}$$

subject to the constraints

$$\mathbb{E}_t f(\boldsymbol{x}^{(t)})_i = 0 \tag{13.9}$$

and

$$\mathbb{E}_t [f(\boldsymbol{x}^{(t)})_i^2] = 1. \tag{13.10}$$