be heads. See Dauphin *et al.* (2014) for a review of the relevant theoretical work.

An amazing property of many random functions is that the eigenvalues of the Hessian become more likely to be positive as we reach regions of lower cost. In our coin tossing analogy, this means we are more likely to have our coin come up heads n times if we are at a critical point with low cost. This means that local minima are much more likely to have low cost than high cost. Critical points with high cost are far more likely to be saddle points. Critical points with extremely high cost are more likely to be local maxima.

This happens for many classes of random functions. Does it happen for neural networks? Baldi and Hornik (1989) showed theoretically that shallow autoencoders (feedforward networks trained to copy their input to their output, described in chapter 14) with no nonlinearities have global minima and saddle points but no local minima with higher cost than the global minimum. They observed without proof that these results extend to deeper networks without nonlinearities. The output of such networks is a linear function of their input, but they are useful to study as a model of nonlinear neural networks because their loss function is a non-convex function of their parameters. Such networks are essentially just multiple matrices composed together. Saxe *et al.* (2013) provided exact solutions to the complete learning dynamics in such networks and showed that learning in these models captures many of the qualitative features observed in the training of deep models with nonlinear activation functions. Dauphin *et al.* (2014) showed experimentally that real neural networks also have loss functions that contain very many high-cost saddle points. Choromanska *et al.* (2014) provided additional theoretical arguments, showing that another class of high-dimensional random functions related to neural networks does so as well.

What are the implications of the proliferation of saddle points for training algorithms? For first-order optimization algorithms that use only gradient information, the situation is unclear. The gradient can often become very small near a saddle point. On the other hand, gradient descent empirically seems to be able to escape saddle points in many cases. Goodfellow *et al.* (2015) provided visualizations of several learning trajectories of state-of-the-art neural networks, with an example given in figure 8.2. These visualizations show a flattening of the cost function near a prominent saddle point where the weights are all zero, but they also show the gradient descent trajectory rapidly escaping this region. Goodfellow *et al.* (2015) also argue that continuous-time gradient descent may be shown analytically to be repelled from, rather than attracted to, a nearby saddle point, but the situation may be different for more realistic uses of gradient descent.

For Newton's method, it is clear that saddle points constitute a problem.