

most commonly used property is the gradient:

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{x}, y \sim \hat{p}_{\text{data}}} \nabla_{\boldsymbol{\theta}} \log p_{\text{model}}(\mathbf{x}, y; \boldsymbol{\theta}). \quad (8.6)$$

Computing this expectation exactly is very expensive because it requires evaluating the model on every example in the entire dataset. In practice, we can compute these expectations by randomly sampling a small number of examples from the dataset, then taking the average over only those examples.

Recall that the standard error of the mean (equation 5.46) estimated from  $n$  samples is given by  $\sigma / \sqrt{n}$ , where  $\sigma$  is the true standard deviation of the value of the samples. The denominator of  $\sqrt{n}$  shows that there are less than linear returns to using more examples to estimate the gradient. Compare two hypothetical estimates of the gradient, one based on 100 examples and another based on 10,000 examples. The latter requires 100 times more computation than the former, but reduces the standard error of the mean only by a factor of 10. Most optimization algorithms converge much faster (in terms of total computation, not in terms of number of updates) if they are allowed to rapidly compute approximate estimates of the gradient rather than slowly computing the exact gradient.

Another consideration motivating statistical estimation of the gradient from a small number of samples is redundancy in the training set. In the worst case, all  $m$  samples in the training set could be identical copies of each other. A sampling-based estimate of the gradient could compute the correct gradient with a single sample, using  $m$  times less computation than the naive approach. In practice, we are unlikely to truly encounter this worst-case situation, but we may find large numbers of examples that all make very similar contributions to the gradient.

Optimization algorithms that use the entire training set are called **batch** or **deterministic** gradient methods, because they process all of the training examples simultaneously in a large batch. This terminology can be somewhat confusing because the word “batch” is also often used to describe the minibatch used by minibatch stochastic gradient descent. Typically the term “batch gradient descent” implies the use of the full training set, while the use of the term “batch” to describe a group of examples does not. For example, it is very common to use the term “batch size” to describe the size of a minibatch.

Optimization algorithms that use only a single example at a time are sometimes called **stochastic** or sometimes **online** methods. The term online is usually reserved for the case where the examples are drawn from a stream of continually created examples rather than from a fixed-size training set over which several passes are made.

Most algorithms used for deep learning fall somewhere in between, using more