$\boldsymbol{H}$ or its inverse amplifies pre-existing errors, in this case, estimation errors in $\boldsymbol{g}$. Very small changes in the estimate of $\boldsymbol{g}$ can thus cause large changes in the update $\boldsymbol{H}^{-1}\boldsymbol{g}$, even if $\boldsymbol{H}$ were estimated perfectly. Of course, $\boldsymbol{H}$ will be estimated only approximately, so the update $\boldsymbol{H}^{-1}\boldsymbol{g}$ will contain even more error than we would predict from applying a poorly conditioned operation to the estimate of $\boldsymbol{g}$.

It is also crucial that the minibatches be selected randomly. Computing an unbiased estimate of the expected gradient from a set of samples requires that those samples be independent. We also wish for two subsequent gradient estimates to be independent from each other, so two subsequent minibatches of examples should also be independent from each other. Many datasets are most naturally arranged in a way where successive examples are highly correlated. For example, we might have a dataset of medical data with a long list of blood sample test results. This list might be arranged so that first we have five blood samples taken at different times from the first patient, then we have three blood samples taken from the second patient, then the blood samples from the third patient, and so on. If we were to draw examples in order from this list, then each of our minibatches would be extremely biased, because it would represent primarily one patient out of the many patients in the dataset. In cases such as these where the order of the dataset holds some significance, it is necessary to shuffle the examples before selecting minibatches. For very large datasets, for example datasets containing billions of examples in a data center, it can be impractical to sample examples truly uniformly at random every time we want to construct a minibatch. Fortunately, in practice it is usually sufficient to shuffle the order of the dataset once and then store it in shuffled fashion. This will impose a fixed set of possible minibatches of consecutive examples that all models trained thereafter will use, and each individual model will be forced to reuse this ordering every time it passes through the training data. However, this deviation from true random selection does not seem to have a significant detrimental effect. Failing to ever shuffle the examples in any way can seriously reduce the effectiveness of the algorithm.

Many optimization problems in machine learning decompose over examples well enough that we can compute entire separate updates over different examples in parallel. In other words, we can compute the update that minimizes $J(\boldsymbol{X})$ for one minibatch of examples $\boldsymbol{X}$ at the same time that we compute the update for several other minibatches. Such asynchronous parallel distributed approaches are discussed further in section 12.1.3.

An interesting motivation for minibatch stochastic gradient descent is that it follows the gradient of the true *generalization error* (equation 8.2) so long as no examples are repeated. Most implementations of minibatch stochastic gradient