specification usually determines the training set and you are not allowed to collect more data.

How can one determine a reasonable level of performance to expect? Typically, in the academic setting, we have some estimate of the error rate that is attainable based on previously published benchmark results. In the real-word setting, we have some idea of the error rate that is necessary for an application to be safe, cost-effective, or appealing to consumers. Once you have determined your realistic desired error rate, your design decisions will be guided by reaching this error rate.

Another important consideration besides the target value of the performance metric is the choice of which metric to use. Several different performance metrics may be used to measure the effectiveness of a complete application that includes machine learning components. These performance metrics are usually different from the cost function used to train the model. As described in section 5.1.2, it is common to measure the accuracy, or equivalently, the error rate, of a system.

However, many applications require more advanced metrics.

Sometimes it is much more costly to make one kind of a mistake than another. For example, an e-mail spam detection system can make two kinds of mistakes: incorrectly classifying a legitimate message as spam, and incorrectly allowing a spam message to appear in the inbox. It is much worse to block a legitimate message than to allow a questionable message to pass through. Rather than measuring the error rate of a spam classifier, we may wish to measure some form of total cost, where the cost of blocking legitimate messages is higher than the cost of allowing spam messages.

Sometimes we wish to train a binary classifier that is intended to detect some rare event. For example, we might design a medical test for a rare disease. Suppose that only one in every million people has this disease. We can easily achieve 99.9999% accuracy on the detection task, by simply hard-coding the classifier to always report that the disease is absent. Clearly, accuracy is a poor way to characterize the performance of such a system. One way to solve this problem is to instead measure **precision** and **recall**. Precision is the fraction of detections reported by the model that were correct, while recall is the fraction of true events that were detected. A detector that says no one has the disease would achieve perfect precision, but zero recall. A detector that says everyone has the disease would achieve perfect recall, but precision equal to the percentage of people who have the disease (0.0001% in our example of a disease that only one people in a million have). When using precision and recall, it is common to plot a **PR curve**, with precision on the y-axis and recall on the x-axis. The classifier generates a score that is higher if the event to be detected occurred. For example, a feedforward