run the hyperparameter optimization methods yourself using the code in the GitHub repo associated with this book.

Hyperparameter Optimization Isn't Just for Deep Networks!

It's worth emphasizing that hyperparameter optimization isn't only for deep networks. Most forms of machine learning algorithms have parameters that can't be learned with the default learning methods. These parameters are also called hyperparameters. You will see some examples of hyperparameters for random forests (another common machine learning method) later in this chapter.

It's worth noting, however, that deep networks tend to be more sensitive to hyperparameter choice than other algorithms. While a random forest might underperform slightly with default choices for hyperparameters, deep networks might fail to learn entirely. For this reason, mastering hyperparameter optimization is a critical skill for a would-be deep learner.

Model Evaluation and Hyperparameter Optimization

In the previous chapters, we have only entered briefly into the question of how to tell whether a machine learning model is good or not. Any measurement of model performance must gauge the model's ability to generalize. That is, can the model make predictions on datapoints it has never seen before? The best test of model performance is to create a model, then evaluate *prospectively* on data that becomes available *after* the model was constructed. However, this sort of test is unwieldy to do regularly. During a design phase, a practicing data scientist may want to evaluate many different types of models or learning algorithms to find which is best.

The solution to this dilemma is to "hold-out" part of the available dataset as a validation set. This validation set will be used to measure the performance of different models (with differing hyperparameter choices). It's also good practice to have a second held-out set, the test set, for gauging the performance of the final model chosen by hyperparameter selection methods.

Let's assume you have a hundred datapoints. A simple procedure would be to use 80 of these datapoints to train prospective models with 20 held-out datapoints used to validate the model choice. The "goodness" of a proposed model can then be tracked by its "score" on the held-out 20 datapoints. Models can be iteratively improved by proposing new designs, and accepting only those that improve performance on the held-out set.

In practice, though, this procedure leads to *overfitting*. Practitioners quickly learn peculiarities of the held-out set and tweak model structure to artificially boost scores on the held-out set. To combat this, practitioners commonly break the held-out set

into two parts: one part for validation of hyperparameters and the other for final model validation. In this case, let's say you reserve 10 datapoints for validation and 10 for final testing. This would be called an 80/10/10 data split.

Why Is the Test Set Necessary?

An important point worth noting is that hyperparameter optimization methods are themselves a form of learning algorithm. In particular, they are a learning algorithm for setting nondifferentiable quantities that aren't easily amenable to calculus-based analysis. The "training set" for the hyperparameter learning algorithm is simply the held-out validation set.

In general, it isn't very meaningful to gauge model performance on their training sets. As always, learned quantities must generalize and it is consequently necessary to test performance on a different set. Since the training set is used for gradient-based learning, and the validation set is used for hyperparameter learning, the test set is necessary to gauge how well learned hyperparameters generalize to new data.

Black-Box Learning Algorithms

Black-box learning algorithms assume no structural information about the systems they are trying to optimize. Most hyperparameter methods are black-box; they work for any type of deep learning or machine learning algorithm.

Black-box methods in general don't scale as well as white-box methods (such as gradient descent) since they tend to get lost in high-dimensional spaces. Due to the lack of directional information from a gradient, black-box methods can get lost in even 50 dimensional spaces (optimizing 50 hyperparameters is quite challenging in practice).

To understand why, suppose there are 50 hyperparameters, each with 3 potential values. Then the black-box algorithm must blindly search a space of size 3^{50}. This can be done, but performing the search will require lots of computational power in general.

Metrics, Metrics, Metrics

When choosing hyperparameters, you want to select those that make the models you design more accurate. In machine learning, a *metric* is a function that gauges the accuracy of predictions from a trained model. Hyperparameter optimization is done to optimize for hyperparameters that maximize (or minimize) this metric on the validation set. While this sounds simple up front, the notion of accuracy can in fact be