Ranking, Recommender Systems, and Other Kinds of Learning

Because this is an introductory book, we focused on the most common machine learning tasks: classification and regression in supervised learning, and clustering and signal decomposition in unsupervised learning. There are many more kinds of machine learning out there, with many important applications. There are two particularly important topics that we did not cover in this book. The first is *ranking*, in which we want to retrieve answers to a particular query, ordered by their relevance. You've probably already used a ranking system today; this is how search engines operate. You input a search query and obtain a sorted list of answers, ranked by how relevant they are. A great introduction to ranking is provided in Manning, Raghavan, and Schütze's book *Introduction to Information Retrieval*. The second topic is *recommender systems*, which provide suggestions to users based on their preferences. You've probably encountered recommender systems under headings like "People You May Know," "Customers Who Bought This Item Also Bought," or "Top Picks for You." There is plenty of literature on the topic, and if you want to dive right in you might be interested in the now classic "Netflix prize challenge", in which the Netflix video streaming site released a large dataset of movie preferences and offered a prize of $1 million to the team that could provide the best recommendations. Another common application is prediction of time series (like stock prices), which also has a whole body of literature devoted to it. There are many more machine learning tasks out there—much more than we can list here—and we encourage you to seek out information from books, research papers, and online communities to find the paradigms that best apply to your situation.

Probabilistic Modeling, Inference, and Probabilistic Programming

Most machine learning packages provide predefined machine learning models that apply one particular algorithm. However, many real-world problems have a particular structure that, when properly incorporated into the model, can yield much better-performing predictions. Often, the structure of a particular problem can be expressed using the language of probability theory. Such structure commonly arises from having a mathematical model of the situation for which you want to predict. To understand what we mean by a structured problem, consider the following example.

Let's say you want to build a mobile application that provides a very detailed position estimate in an outdoor space, to help users navigate a historical site. A mobile phone provides many sensors to help you get precise location measurements, like the GPS, accelerometer, and compass. You also have an exact map of the area. This problem is highly structured. You know where the paths and points of interest are from your map. You also have rough positions from the GPS, and the accelerometer and compass in the user's device provide you with very precise relative measurements. But throwing these all together into a black-box machine learning system to predict positions might not be the best idea. This would throw away all the information you