1 Introduction

This chapter presents a preliminary introduction to machine learning, including an overview of some key learning tasks and applications, basic definitions and terminology, and the discussion of some general scenarios.

1.1 What is machine learning?

Machine learning can be broadly defined as computational methods using experience to improve performance or to make accurate predictions. Here, *experience* refers to the past information available to the learner, which typically takes the form of electronic data collected and made available for analysis. This data could be in the form of digitized human-labeled training sets, or other types of information obtained via interaction with the environment. In all cases, its quality and size are crucial to the success of the predictions made by the learner.

An example of a learning problem is how to use a finite sample of randomly selected documents, each labeled with a topic, to accurately predict the topic of unseen documents. Clearly, the larger is the sample, the easier is the task. But the difficulty of the task also depends on the quality of the labels assigned to the documents in the sample, since the labels may not be all correct, and on the number of possible topics.

Machine learning consists of designing efficient and accurate prediction *algorithms*. As in other areas of computer science, some critical measures of the quality of these algorithms are their time and space complexity. But, in machine learning, we will need additionally a notion of *sample complexity* to evaluate the sample size required for the algorithm to learn a family of concepts. More generally, theoretical learning guarantees for an algorithm depend on the complexity of the concept classes considered and the size of the training sample.

Since the success of a learning algorithm depends on the data used, machine learning is inherently related to data analysis and statistics. More generally, learning techniques are data-driven methods combining fundamental concepts in computer science with ideas from statistics, probability and optimization.

1.2 What kind of problems can be tackled using machine learning?

Predicting the label of a document, also known as document classification, is by no means the only learning task. Machine learning admits a very broad set of practical applications, which include the following:

- Text or document classification. This includes problems such as assigning a topic to a text or a document, or determining automatically if the content of a web page is inappropriate or too explicit; it also includes spam detection.
- Natural language processing (NLP). Most tasks in this field, including part-of-speech tagging, named-entity recognition, context-free parsing, or dependency parsing, are cast as learning problems. In these problems, predictions admit some structure. For example, in part-of-speech tagging, the prediction for a sentence is a sequence of part-of-speech tags labeling each word. In context-free parsing the prediction is a tree. These are instances of richer learning problems known as *structured prediction problems*.
- Speech processing applications. This includes speech recognition, speech synthesis, speaker verification, speaker identification, as well as sub-problems such as language modeling and acoustic modeling.
- Computer vision applications. This includes object recognition, object identification, face detection, Optical character recognition (OCR), content-based image retrieval, or pose estimation.
- Computational biology applications. This includes protein function prediction, identification of key sites, or the analysis of gene and protein networks.
- Many other problems such as fraud detection for credit card, telephone or insurance companies, network intrusion, learning to play games such as chess, backgammon, or Go, unassisted control of vehicles such as robots or cars, medical diagnosis, the design of recommendation systems, search engines, or information extraction systems, are tackled using machine learning techniques.

This list is by no means comprehensive. Most prediction problems found in practice can be cast as learning problems and the practical application area of machine learning keeps expanding. The algorithms and techniques discussed in this book can be used to derive solutions for all of these problems, though we will not discuss in detail these applications.

1.3 Some standard learning tasks

The following are some standard machine learning tasks that have been extensively studied:

- *Classification*: this is the problem of assigning a category to each item. For example, document classification consists of assigning a category such as *politics*, *business*, *sports*, or *weather* to each document, while image classification consists of assigning to each image a category such as *car*, *train*, or *plane*. The number of categories in such tasks is often less than a few hundreds, but it can be much larger in some difficult tasks and even unbounded as in OCR, text classification, or speech recognition.
- *Regression*: this is the problem of predicting a real value for each item. Examples of regression include prediction of stock values or that of variations of economic variables. In regression, the penalty for an incorrect prediction depends on the magnitude of the difference between the true and predicted values, in contrast with the classification problem, where there is typically no notion of closeness between various categories.
- *Ranking*: this is the problem of learning to order items according to some criterion. Web search, e.g., returning web pages relevant to a search query, is the canonical ranking example. Many other similar ranking problems arise in the context of the design of information extraction or natural language processing systems.
- *Clustering*: this is the problem of partitioning a set of items into homogeneous subsets. Clustering is often used to analyze very large data sets. For example, in the context of social network analysis, clustering algorithms attempt to identify natural *communities* within large groups of people.
- *Dimensionality reduction* or *manifold learning*: this problem consists of transforming an initial representation of items into a lower-dimensional representation while preserving some properties of the initial representation. A common example involves preprocessing digital images in computer vision tasks.

The main practical objectives of machine learning consist of generating accurate predictions for unseen items and of designing efficient and robust algorithms to produce these predictions, even for large-scale problems. To do so, a number of algorithmic and theoretical questions arise. Some fundamental questions include: Which concept families can actually be learned, and under what conditions? How well can these concepts be learned computationally?

1.4 Learning stages

Here, we will use the canonical problem of spam detection as a running example to illustrate some basic definitions and describe the use and evaluation of machine learning algorithms in practice, including their different stages.

Spam detection is the problem of learning to automatically classify email messages as either SPAM or non-SPAM. The following is a list of definitions and terminology commonly used in machine learning:

- *Examples*: Items or instances of data used for learning or evaluation. In our spam problem, these examples correspond to the collection of email messages we will use for learning and testing.
- *Features*: The set of attributes, often represented as a vector, associated to an example. In the case of email messages, some relevant features may include the length of the message, the name of the sender, various characteristics of the header, the presence of certain keywords in the body of the message, and so on.
- *Labels*: Values or categories assigned to examples. In classification problems, examples are assigned specific categories, for instance, the SPAM and non-SPAM categories in our binary classification problem. In regression, items are assigned real-valued labels.
- *Hyperparameters*: Free parameters that are not determined by the learning algorithm, but rather specified as inputs to the learning algorithm.
- *Training sample*: Examples used to train a learning algorithm. In our spam problem, the training sample consists of a set of email examples along with their associated labels. The training sample varies for different learning scenarios, as described in section 1.5.
- *Validation sample*: Examples used to tune the parameters of a learning algorithm when working with labeled data. The validation sample is used to select appropriate values for the learning algorithm's free parameters (hyperparameters).
- *Test sample*: Examples used to evaluate the performance of a learning algorithm. The test sample is separate from the training and validation data and is not made available in the learning stage. In the spam problem, the test sample consists of a collection of email examples for which the learning algorithm must predict labels based on features. These predictions are then compared with the labels of the test sample to measure the performance of the algorithm.
- *Loss function*: A function that measures the difference, or loss, between a predicted label and a true label. Denoting the set of all labels as $\mathcal{Y}$ and the set of possible predictions as $\mathcal{Y}'$, a loss function L is a mapping $L\colon \mathcal{Y} \times \mathcal{Y}' \to \mathbb{R}_+$. In most cases, $\mathcal{Y}' = \mathcal{Y}$ and the loss function is bounded, but these conditions do not always hold. Common examples of loss functions include the zero-one (or

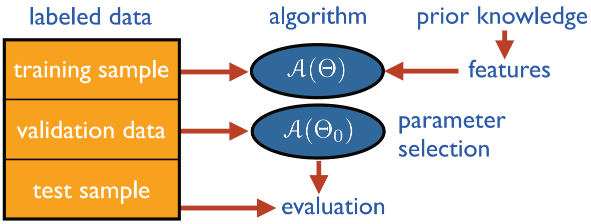

Figure 1.1
Illustration of the typical stages of a learning process.

misclassification) loss defined over $\{-1,+1\} \times \{-1,+1\}$ by $L(y, y') = 1_{y' \neq y}$ and the squared loss defined over $\mathcal{I} \times \mathcal{I}$ by $L(y, y') = (y' - y)^2$, where $\mathcal{I} \subseteq \mathbb{R}$ is typically a bounded interval.

- *Hypothesis set*: A set of functions mapping features (feature vectors) to the set of labels $\mathcal{Y}$. In our example, these may be a set of functions mapping email features to $\mathcal{Y} = \{\text{SPAM}, \text{non-SPAM}\}$. More generally, hypotheses may be functions mapping features to a different set $\mathcal{Y}'$. They could be linear functions mapping email feature vectors to real numbers interpreted as *scores* ($\mathcal{Y}' = \mathbb{R}$), with higher score values more indicative of SPAM than lower ones.

We now define the learning stages of our spam problem (see figure 1.1). We start with a given collection of labeled examples. We first randomly partition the data into a training sample, a validation sample, and a test sample. The size of each of these samples depends on a number of different considerations. For example, the amount of data reserved for validation depends on the number of hyperparameters of the algorithm, which are represented here by the vector Θ. Also, when the labeled sample is relatively small, the amount of training data is often chosen to be larger than that of the test data since the learning performance directly depends on the training sample.

Next, we associate relevant features to the examples. This is a critical step in the design of machine learning solutions. Useful features can effectively guide the learning algorithm, while poor or uninformative ones can be misleading. Although it is critical, to a large extent, the choice of the features is left to the user. This choice reflects the user's *prior knowledge* about the learning task which in practice can have a dramatic effect on the performance results.

Now, we use the features selected to train our learning algorithm $\mathcal{A}$ by tuning the values of its free parameters Θ (also called *hyperparameters*). For each value of these parameters, the algorithm selects a different hypothesis out of the hypothesis set. We choose the one resulting in the best performance on the validation sample (Θ_0). Finally, using that hypothesis, we predict the labels of the examples in the test sample. The performance of the algorithm is evaluated by using the loss function associated to the task, e.g., the zero-one loss in our spam detection task, to compare the predicted and true labels. Thus, the performance of an algorithm is of course evaluated based on its test error and not its error on the training sample.

1.5 Learning scenarios

We next briefly describe some common machine learning scenarios. These scenarios differ in the types of training data available to the learner, the order and method by which training data is received and the test data used to evaluate the learning algorithm.

- *Supervised learning*: The learner receives a set of labeled examples as training data and makes predictions for all unseen points. This is the most common scenario associated with classification, regression, and ranking problems. The spam detection problem discussed in the previous section is an instance of supervised learning.
- *Unsupervised learning*: The learner exclusively receives unlabeled training data, and makes predictions for all unseen points. Since in general no labeled example is available in that setting, it can be difficult to quantitatively evaluate the performance of a learner. Clustering and dimensionality reduction are example of unsupervised learning problems.
- *Semi-supervised learning*: The learner receives a training sample consisting of both labeled and unlabeled data, and makes predictions for all unseen points. Semi-supervised learning is common in settings where unlabeled data is easily accessible but labels are expensive to obtain. Various types of problems arising in applications, including classification, regression, or ranking tasks, can be framed as instances of semi-supervised learning. The hope is that the distribution of unlabeled data accessible to the learner can help him achieve a better performance than in the supervised setting. The analysis of the conditions under which this can indeed be realized is the topic of much modern theoretical and applied machine learning research.
- *Transductive inference*: As in the semi-supervised scenario, the learner receives a labeled training sample along with a set of unlabeled test points. However, the objective of transductive inference is to predict labels only for these particular test points. Transductive inference appears to be an easier task and matches the scenario encountered in a variety of modern applications. However, as in the semi-supervised setting, the assumptions under which a better performance can be achieved in this setting are research questions that have not been fully resolved.

- *On-line learning*: In contrast with the previous scenarios, the online scenario involves multiple rounds where training and testing phases are intermixed. At each round, the learner receives an unlabeled training point, makes a prediction, receives the true label, and incurs a loss. The objective in the on-line setting is to minimize the cumulative loss over all rounds or to minimize the *regret*, that is the difference of the cumulative loss incurred and that of the best expert in hindsight. Unlike the previous settings just discussed, no distributional assumption is made in on-line learning. In fact, instances and their labels may be chosen adversarially within this scenario.
- *Reinforcement learning*: The training and testing phases are also intermixed in reinforcement learning. To collect information, the learner actively interacts with the environment and in some cases affects the environment, and receives an immediate reward for each action. The object of the learner is to maximize his reward over a course of actions and iterations with the environment. However, no long-term reward feedback is provided by the environment, and the learner is faced with the *exploration versus exploitation* dilemma, since he must choose between exploring unknown actions to gain more information versus exploiting the information already collected.
- *Active learning*: The learner adaptively or interactively collects training examples, typically by querying an oracle to request labels for new points. The goal in active learning is to achieve a performance comparable to the standard supervised learning scenario (or *passive learning* scenario), but with fewer labeled examples. Active learning is often used in applications where labels are expensive to obtain, for example computational biology applications.

In practice, many other intermediate and somewhat more complex learning scenarios may be encountered.

1.6 Generalization

Machine learning is fundamentally about *generalization*. As an example, the standard supervised learning scenario consists of using a finite sample of labeled examples to make accurate predictions about unseen examples. The problem is typically formulated as that of selecting a function out of a *hypothesis set*, that is a subset of the family of all functions. The function selected is subsequently used to label all instances, including unseen examples.

How should a hypothesis set be chosen? With a rich or *complex* hypothesis set, the learner may choose a function or predictor that is *consistent* with the training sample, that is one that commits no error on the training sample. With a less complex family, incurring some errors on the training sample may be unavoidable. But,

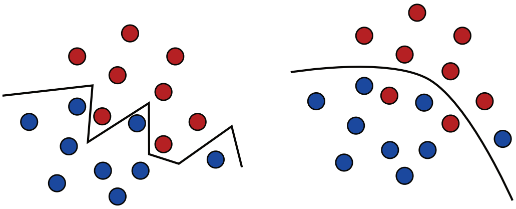

Figure 1.2
The zig-zag line on the left panel is consistent over the blue and red training sample, but it is a complex separation surface that is not likely to generalize well to unseen data. In contrast, the decision surface on the right panel is simpler and might generalize better in spite of its misclassification of a few points of the training sample.

which will lead to a better generalization? How should we define the complexity of a hypothesis set?

Figure 1.2 illustrates these two types of solution: one is a zig-zag line that perfectly separates the two populations of blue and red points and that is chosen from a complex family; the other one is a smoother line chosen from a simpler family that only imperfectly discriminates between the two sets. We will see that, in general, the best predictor on the training sample may not be the best overall. A predictor chosen from a very complex family can essentially memorize the data, but generalization is distinct from the memorization of the training labels.

We will see that the trade-off between the sample size and complexity plays a critical role in generalization. When the sample size is relatively small, choosing from a too complex a family may lead to poor generalization, which is also known as *overfitting*. On the other hand, with a too simple a family it may not be possible to achieve a sufficient accuracy, which is known as *underfitting*.

In the next chapters, we will analyze more in detail the problem of generalization and will seek to derive theoretical guarantees for learning. This will depend on different notions of complexity that we will thoroughly discuss.