Chapter 1

Learning from Data

The aim of this course is to introduce the fundamentals of machine learning. Our goal is to cover key fundamental topics, with an emphasis on theoretical and conceptual foundations. We will *not* cover every popular machine learning algorithm. We will—however—cover the key principles that you need in order to understand every popular machine learning algorithm. For this reason, this course is organized around key concepts—such as decision boundaries, likelihood-based modeling, and information theory—rather than individual methods. In this chapter, we begin with a brief and high-level overview of some key concepts and definitions, which re-occur throughout all the chapters of this course.

1.1 What is Machine Learning?

There is no canonical definition of machine learning. If you Google the definition, you will get the following text, from Oxford Languages:

> The use and development of computer systems that are able to learn and adapt without following explicit instructions, by using algorithms and statistical models to analyze and draw inferences from patterns in data.

The key point is that we have some kind of *computer system*, which is able to *learn, adapt, and draw inferences* from data. This is a broad reaching definition, but machine learning encompasses a very wide range of methods.

Machine learning in the real world In the past decade, machine learning has become more and more prevalent in our everyday lives. Some of the most obvious examples of machine learning impacting our day-to-day lives include

- *content recommendation* (e.g., used by Google and Facebook),
- *text classification* (e.g., to classify spam emails),
- *image classification* (e.g., for automated diagnoses from x-rays),

- *text generation* (e.g., for automated translation and dialogue),
- *time-series forecasting* (e.g., used by hedge funds to pick stocks).

In this course, we will often focus on machine learning as an abstract problem, without focusing on a particular real-world use case. However, it is critically important that we keep these use cases in mind, both as a motivation and also to ensure that we consider the ethical grounding of our algorithms.

1.2 Datasets, Features and Targets

The one thing that every machine learning application has in common is data. Indeed, the key factor distinguishing machine learning from more general forms of artificial intelligence (AI)—such as *expert systems*—is that machine learning models learn from data, rather than relying on pre-defined rules.[1]

The core of any machine learning application, and the single thing that we will always assume, is access to a dataset $\mathcal{D}$. In most settings, we assume that this dataset consists of a set of examples $\mathcal{D} = \{(\mathbf{x}_i, y_i), i = 1...n\}$. The $\mathbf{x} \in \mathcal{X}$ values are often called *features*, while the $y \in \mathcal{Y}$ are usually called *targets or labels*. The goal of machine learning—generally speaking—is to learn a mapping

$$f : \mathcal{X} \to \mathcal{Y} \tag{1.1}$$

from features to targets, based on the training data. The features are typically represented as vectors, whereas the target is usually a single value. Conceptually, the features are the information that we are using to make the prediction, while the target label is what we are trying to predict.

Accuracy and loss

Typically, the goal is to find a function that is able to approximate the feature-to-target mapping that is implicit in our dataset, i.e., we want

$$f(\mathbf{x}) \approx y, \qquad \forall (\mathbf{x}, y) \in \mathcal{D}. \tag{1.2}$$

In many cases, we have some error or *loss metric* $L : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$, which measures the difference between a model prediction $\hat{y} = f(\mathbf{x})$ and a true value y. The goal is to find a function that gives a small (or minimal) loss on our training data. Alternatively, we can consider *accuracy metrics*, which measure how close our model's output is to the true value. When using an accuracy metric, we are trying to find a model that maximizes the accuracy.

[1] As we will see—however—there is plenty of hand-crafting and rule-like methods that go into modern machine learning models.

For instance, the simplest loss metric—called the 0-1 loss—will give a value of 1 if the prediction is incorrect and 0 otherwise:

$$L_{0\text{-}1}(y, \hat{y}) = \begin{cases} 1 & \text{if } y \neq \hat{y} \\ 0 & \text{otherwise} \end{cases} \tag{1.3}$$

The simplest accuracy metric is just the complement of the 0-1 loss: we get a 1 for making correct predictions and a 0 otherwise. Often we will average our loss or accuracy over the dataset, e.g., to get the percentage accuracy.

Binary, categorical, and real values

The features and targets in our machine learning model can be real valued, categorical (i.e., discrete), or binary. This is why we used the generic domain $\mathcal{X}$ for the features and $\mathcal{Y}$ for the targets in the equations above. The nature of the target has a major influence on the task. In the case where the target is binary (i.e., $y \in \{0, 1\}$) or discrete (i.e., $y \in \mathbb{Z}$), we typically refer to the learning task as *classification*, since we are classifying among a discrete set of categories. We refer to cases where the target is real-valued (i.e., $y \in \mathbb{R}$) as *regression* problems.

The nature of the features also important, but it does not have such a strong influence on algorithm design. It is possible to mix and match features of different types within a single problem, and often we will see problems that involve both real-valued and categorical features. Note also that the different types of features are special cases of one another, in a mathematical sense: categorical (i.e., discrete) values (e.g., integers) are a subset of the full space of real numbers (i.e., $\mathbb{Z} \subset \mathbb{R}$), while binary 0/1 values are a subset of the integers (i.e., $\{0, 1\} \subset \mathbb{Z}$). For this reason, we will often assume that features are real-valued without a loss of generality.

Example 1. Spam classification. *In the example of spam classification, our target is a binary variable (i.e., whether the email is spam). The different dimensions of the feature vector* $\mathbf{x}$ *might correspond to binary indicators regarding the content of the email: one feature might indicate the presence of all caps; another feature might indicate the presence of of an attachment; etc (Figure 1.1).*

Example 2. Automated diagnostics. *Another example of a classification task might involve classifying images of skin lesions to determine if they are benign or malignant tumors. In this case, the output y might be a categorical variable (e.g., classify as "benign", "malignant", or "no lesion"). The different dimensions of the feature vector* $\mathbf{x}$ *might correspond to properties of the tumor, which have been extracted from the image via pre-processing, such as its size (real-valued) or its texture (categorical). Alternatively, we might consider using the raw image as an input feature vector* $\mathbf{x}$.[2]

Example 3. Stock price prediction. *Predicting the price of a stock would be an example of a regression task, since the target would be real-valued in this case.*

[2]Note that the image is technically a matrix, but we can simply reshape it a vector.

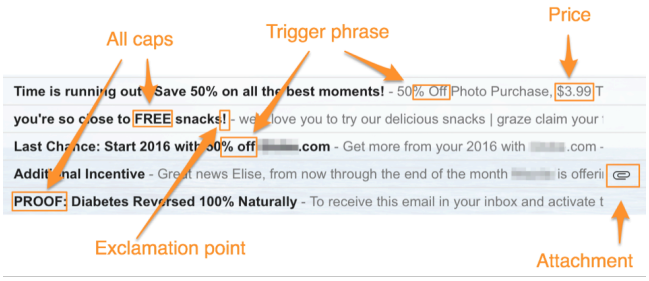

Figure 1.1: Example of features that we might extract for spam classification.

The features for stock price prediction might be a mix of binary, categorical, and real-valued: we might use historical stock price values (real-valued), indicators for mentions of the stock on social media (binary), or ratings from analysts (categorical).

1.3 Testing Sets and Generalization

Generalization is a critical aspect of machine learning. We do not want a model that can only achieve good performance a small sample of datapoints. We want a model that can achieve strong performance on many datapoints and *even datapoints that we have not seen before.* For this reason, in machine learning we always separate our data into disjoint sets: a training set $\mathcal{D}_{\text{trn}}$ and testing set $\mathcal{D}_{\text{tst}}$. We use the training set to train or optimize our model, but we use the testing set to measure our model's performance. The critical point is that the the only time we ever get to use the test set is when we evaluate the *final* performance of our model.

Why is this separation of training and testing important? Well, if we don't do this, then our model can achieve strong performance on the training set simply by *memorizing* the training data. In other words, we could define our prediction function f to simply be a lookup table, where[3]

$$f(\mathbf{x}) = \begin{cases} y_i & \text{if } \exists(\mathbf{x}_i, y_i) \in \mathcal{D}_{\text{trn}} : \mathbf{x} = \mathbf{x}_i \\ \text{undefined} & \text{otherwise.} \end{cases} \tag{1.4}$$

This function would always achieve perfect accuracy (or zero loss) on the training set. However, if we run this function on new datapoints from the test set, its predictions will be undefined. A model that simply memorizes the training data is useless in the real world. We want models that can make useful predictions on points they have not seen before.

Throughout this course, we will develop more refined and formal definitions of memorization and generalization. At this point, however, it is critical to

[3]Note that this assumes all the training points are unique.

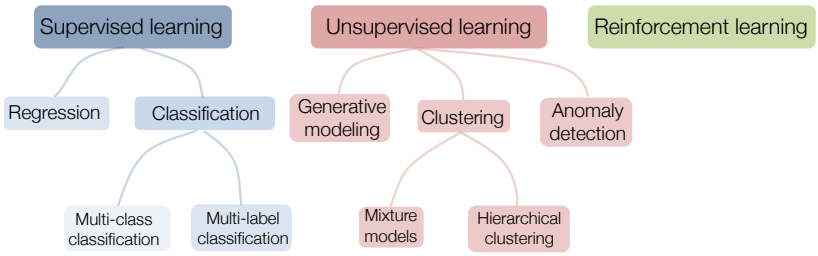

Figure 1.2: A rough taxonomy of some popular machine learning tasks.

understand that we must always hold out a portion of the data as a hidden test set, which is only used for evaluation.

1.4 Supervised and Unsupervised Learning

In the previous sections, we discussed machine learning tasks where the goal is to infer a target label from some features. We also assumed that we were given a training set containing known feature-target pairs. Strictly speaking, this form of machine learning—where we are given labelled training examples—is known as *supervised* machine learning. We call it supervised because the model is given explicit feedback about the kinds of errors that it is making. Supervised tasks are usually distinguished based on whether we have real-valued or categorical features, using the terms regression and classification—respectively—for these two different tasks. In addition, in the case of classification, we often distinguish between tasks where every point has a single label (i.e., multi-class classification) and tasks where each point might have multiple labels (i.e., multi-label classification).

Another popular setting for machine learning is *unsupervised* learning. In unsupervised learning, we are only given access to the features and no labels for our datapoints. The goal of unsupervised learning techniques is to uncover useful structure in the data, including tasks such as generative modeling (i.e., learning to generate data), clustering, and detecting anomalies. We will cover a number of unsupervised methods in the latter portion of this course.

Lastly, beyond unsupervised and supervised learning, there are other more exotic forms of machine learning. This includes the popular task of *reinforcement learning*, where we must learn to make sequential decisions based on delayed feedback. There are also variants of machine learning that border between these different areas, such as semi-supervised learning (which combines unsupervised and supervised learning) and structured prediction (which combines reinforcement learning and supervised learning). These more exotic forms of machine learning will not be covered in this course, but are the focus of many graduate-level machine learning courses.