1 Introduction to Machine Learning

1.1 Introduction

Machine learning is a unified algorithmic framework designed to identify computational models that accurately describe empirical data and the phenomena underlying it, with little or no human involvement. While still a young discipline with much more awaiting discovery than is currently known, today machine learning can be used to teach computers to perform a wide array of useful tasks including automatic detection of objects in images (a crucial component of driver-assisted and self-driving cars), speech recognition (which powers voice command technology), knowledge discovery in the medical sciences (used to improve our understanding of complex diseases), and predictive analytics (leveraged for sales and economic forecasting), to just name a few.

In this chapter we give a high-level introduction to the field of machine learning as well as the contents of this textbook.

1.2 Distinguishing Cats from Dogs: a Machine Learning Approach

To get a big-picture sense of how machine learning works, we begin by discussing a toy problem: teaching a computer how to distinguish between pictures of *cats* from those with *dogs*. This will allow us to informally describe the terminology and procedures involved in solving the typical machine learning problem.

Do you recall how you first learned about the difference between cats and dogs, and how they are different animals? The answer is probably no, as most humans learn to perform simple cognitive tasks like this very early on in the course of their lives. One thing is certain, however: young children do not need some kind of formal scientific training, or a zoological lecture on *felis catus* and *canis familiaris* species, in order to be able to tell cats and dogs apart. Instead, they learn by example. They are naturally presented with many images of what they are told by a *supervisor* (a parent, a caregiver, etc.) are either cats or dogs, until they fully grasp the two concepts. How do we know when a child can successfully distinguish between cats and dogs? Intuitively, when

they encounter new (images of) cats and dogs, and can correctly identify each new example or, in other words, when they can *generalize* what they have learned to new, previously unseen, examples.

Like human beings, computers can be taught how to perform this sort of task in a similar manner. This kind of task where we aim to teach a computer to distinguish between different types or *classes* of things (here *cats* and *dogs*) is referred to as a *classification* problem in the jargon of machine learning, and is done through a series of steps which we detail below.

1. Data collection. Like human beings, a computer must be trained to recognize the difference between these two types of animals by learning from a batch of examples, typically referred to as a *training set* of data. Figure 1.1 shows such a training set consisting of a few images of different cats and dogs. Intuitively, the larger and more diverse the training set the better a computer (or human) can perform a learning task, since exposure to a wider breadth of examples gives the learner more experience.

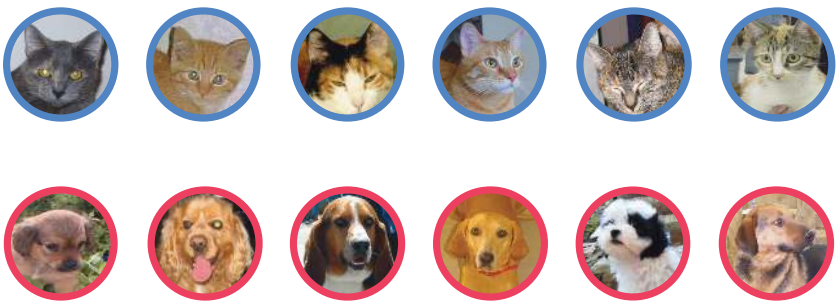

Figure 1.1 A training set consisting of six images of cats (highlighted in blue) and six images of dogs (highlighted in red). This set is used to train a machine learning model that can distinguish between future images of cats and dogs. The images in this figure were taken from [1].

2. Feature design. Think for a moment about how we (humans) tell the difference between images containing cats from those containing dogs. We use color, size, the shape of the ears or nose, and/or some combination of these *features* in order to distinguish between the two. In other words, we do not just look at an image as simply a collection of many small square pixels. We pick out grosser details, or features, from images like these in order to identify what it is that we are looking at. This is true for computers as well. In order to successfully train a computer to perform this task (and any machine learning task more generally)

we need to provide it with properly designed features or, ideally, have it find or *learn* such features itself.

Designing quality features is typically not a trivial task as it can be very application dependent. For instance, a feature like *color* would be less helpful in discriminating between cats and dogs (since many cats and dogs share similar hair colors) than it would be in telling grizzly bears and polar bears apart! Moreover, extracting the features from a training dataset can also be challenging. For example, if some of our training images were blurry or taken from a perspective where we could not see the animal properly, the features we designed might not be properly extracted.

However, for the sake of simplicity with our toy problem here, suppose we can easily extract the following two features from each image in the training set: *size of nose* relative to the size of the head, ranging from small to large, and *shape of ears*, ranging from round to pointy.

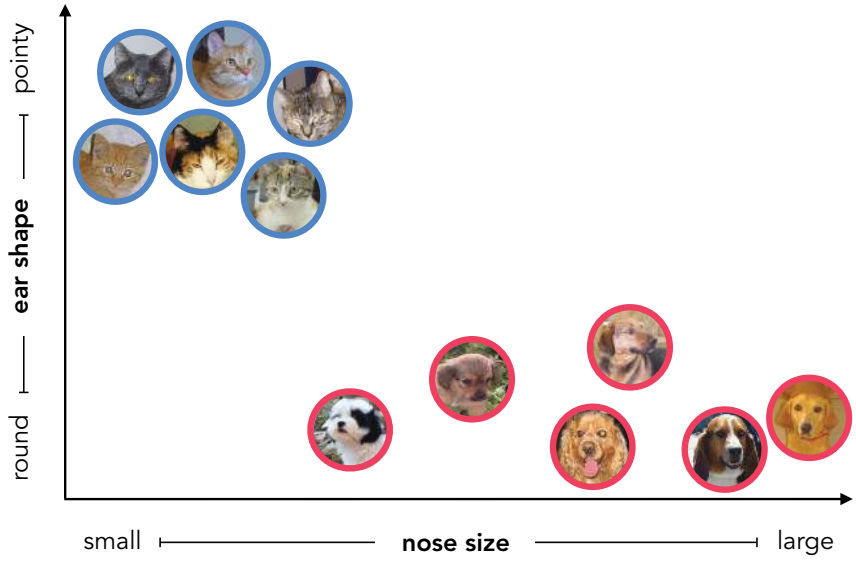

Figure 1.2 Feature space representation of the training set shown in Figure 1.1 where the horizontal and vertical axes represent the features *nose size* and *ear shape*, respectively. The fact that the cats and dogs from our training set lie in distinct regions of the feature space reflects a good choice of features.

Examining the training images shown in Figure 1.1, we can see that all cats have small noses and pointy ears, while dogs generally have large noses and round ears. Notice that with the current choice of features each image can now be represented by just two numbers: a number expressing the relative nose size, and another number capturing the pointiness or roundness of the ears. In other words, we can represent each image in our training set in a two-dimensional

feature space where the features *nose size* and *ear shape* are the horizontal and vertical coordinate axes, respectively, as illustrated in Figure 1.2.

3. Model training. With our feature representation of the training data the machine learning problem of distinguishing between cats and dogs is now a simple geometric one: have the machine find a line or a curve that separates the cats from the dogs in our carefully designed feature space. Supposing for simplicity that we use a line, we must find the right values for its two parameters – a slope and vertical intercept – that define the line's orientation in the feature space. The process of determining proper parameters relies on a set of tools known as *mathematical optimization* detailed in Chapters 2 through 4 of this text, and the tuning of such a set of parameters to a training set is referred to as the training of a model.

Figure 1.3 shows a trained linear model (in black) which divides the feature space into cat and dog regions. This linear model provides a simple computational rule for distinguishing between cats and dogs: when the feature representation of a future image lies above the line (in the blue region) it will be considered a cat by the machine, and likewise any representation that falls below the line (in the red region) will be considered a dog.

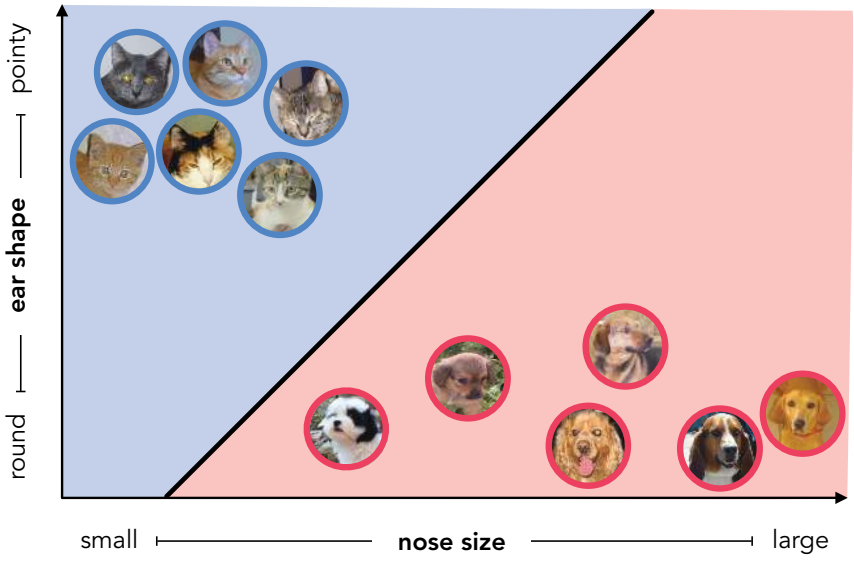

Figure 1.3 A trained linear model (shown in black) provides a computational rule for distinguishing between cats and dogs. Any new image received in the future will be classified as a cat if its feature representation lies above this line (in the blue region), and a dog if the feature representation lies below this line (in the red region).

Figure 1.4 A validation set of cat and dog images (also taken from [1]). Notice that the images in this set are not highlighted in red or blue (as was the case with the training set shown in Figure 1.1) indicating that the true identity of each image is not revealed to the learner. Notice that one of the dogs, the Boston terrier in the bottom right corner, has both a small nose and pointy ears. Because of our chosen feature representation the computer will think this is a cat!

4. Model validation. To validate the efficacy of our trained learner we now show the computer a batch of previously unseen images of cats and dogs, referred to generally as a *validation set* of data, and see how well it can identify the animal in each image. In Figure 1.4 we show a sample validation set for the problem at hand, consisting of three new cat and dog images. To do this, we take each new image, extract our designed features (i.e., nose size and ear shape), and simply check which side of our line (or *classifier*) the feature representation falls on. In this instance, as can be seen in Figure 1.5, all of the new cats and all but one dog from the validation set have been identified correctly by our trained model.

The misidentification of the single dog (a Boston terrier) is largely the result of our choice of features, which we designed based on the training set in Figure 1.1, and to some extent our decision to use a *linear* model (instead of a *nonlinear* one). This dog has been misidentified simply because its features, a small nose and pointy ears, match those of the cats from our training set. Therefore, while it first appeared that a combination of nose size and ear shape could indeed distinguish cats from dogs, we now see through validation that our training set was perhaps too small and not diverse enough for this choice of features to be completely effective in general.

We can take a number of steps to improve our learner. First and foremost we should collect more data, forming a larger and more diverse training set. Second, we can consider designing/including more discriminating features (perhaps eye color, tail shape, etc.) that further help distinguish cats from dogs using a linear model. Finally, we can also try out (i.e., train and validate) an array of *nonlinear* models with the hopes that a more complex rule might better distinguish between cats and dogs. Figure 1.6 compactly summarizes the four steps involved in solving our toy cat-versus-dog classification problem.

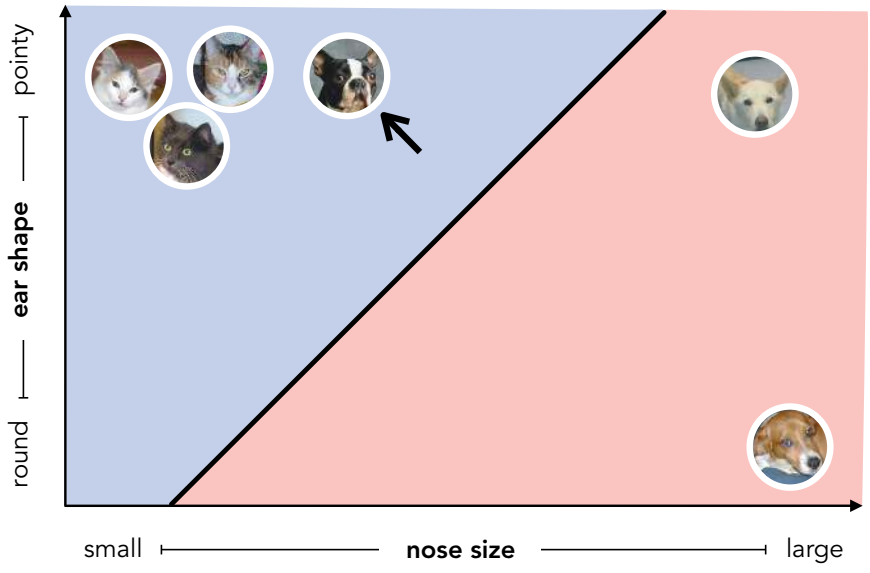

Figure 1.5 Identification of (the feature representation of) validation images using our trained linear model. The Boston terrier (pointed to by an arrow) is *misclassified* as a cat since it has pointy ears and a small nose, just like the cats in our training set.

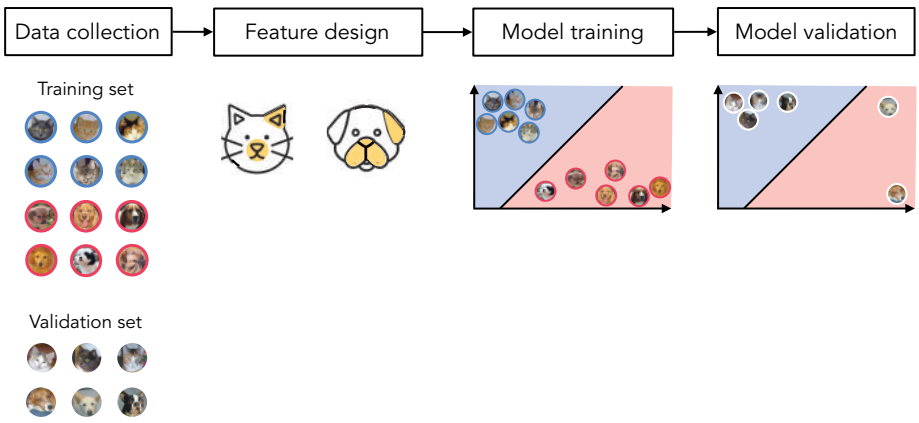

Figure 1.6 The schematic pipeline of our toy cat-versus-dog classification problem. The same general pipeline is used for essentially all machine learning problems.

1.3 The Basic Taxonomy of Machine Learning Problems

The sort of computational rules we can learn using machine learning generally fall into two main categories called *supervised* and *unsupervised* learning, which we discuss next.

1.3.1 Supervised learning

Supervised learning problems (like the prototypical problem outlined in Section 1.2) refer to the automatic learning of computational rules involving input/output relationships. Applicable to a wide array of situations and data types, this type of problem comes in two forms, called *regression* and *classification*, depending on the general numerical form of the output.

Regression

Suppose we wanted to predict the share price of a company that is about to go public. Following the pipeline discussed in Section 1.2, we first gather a training set of data consisting of a number of corporations (preferably active in the same domain) with known share prices. Next, we need to design feature(s) that are thought to be relevant to the task at hand. The company's revenue is one such potential feature, as we can expect that the higher the revenue the more expensive a share of stock should be. To connect the share price (output) to the revenue (input) we can train a simple linear model or *regression line* using our training data.

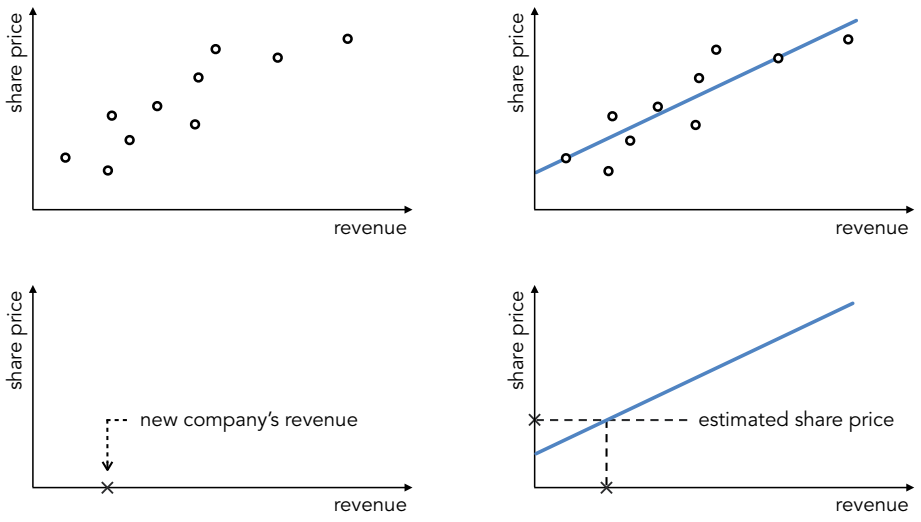

Figure 1.7 (top-left panel) A toy training dataset consisting of ten corporations' share price and revenue values. (top-right panel) A linear model is fit to the data. This trend line models the overall trajectory of the points and can be used for prediction in the future as shown in the bottom-left and bottom-right panels.

The top panels of Figure 1.7 show a toy dataset comprising share price versus revenue information for ten companies, as well as a linear model fit to this data. Once the model is trained, the share price of a new company can be predicted

based on its revenue, as depicted in the bottom panels of this figure. Finally, comparing the predicted price to the actual price for a validation set of data we can test the performance of our linear regression model and apply changes as needed, for example, designing new features (e.g., total assets, total equity, number of employees, years active, etc.) and/or trying more complex nonlinear models.

This sort of task, i.e., fitting a model to a set of training data so that predictions about a *continuous-valued* output (here, share price) can be made, is referred to as regression. We begin our detailed discussion of regression in Chapter 5 with the linear case, and move to nonlinear models starting in Chapter 10 and throughout Chapters 11–14. Below we describe several additional examples of regression to help solidify this concept.

Example 1.1 The rise of student loan debt in the United States

Figure 1.8 (data taken from [2]) shows the total student loan debt (that is money borrowed by students to pay for college tuition, room and board, etc.) held by citizens of the United States from 2006 to 2014, measured quarterly. Over the eight-year period reflected in this plot the student debt has nearly tripled, totaling over one trillion dollars by the end of 2014. The regression line (in black) fits this dataset quite well and, with its sharp positive slope, emphasizes the point that student debt is rising dangerously fast. Moreover, if this trend continues, we can use the regression line to predict that total student debt will surpass two trillion dollars by the year 2026 (we revisit this problem later in Exercise 5.1).

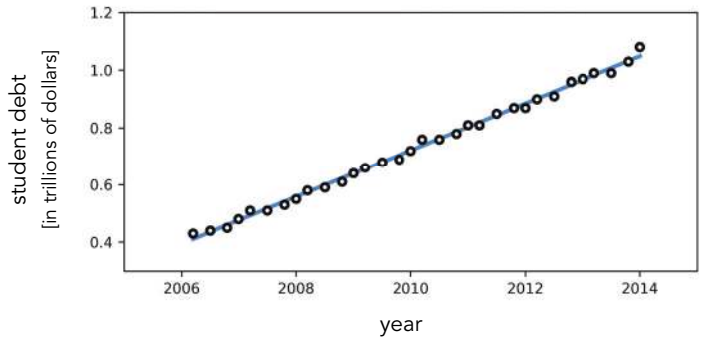

Figure 1.8 Figure associated with Example 1.1, illustrating total student loan debt in the United States measured quarterly from 2006 to 2014. The rapid increase rate of the debt, measured by the slope of the trend line fit to the data, confirms that student debt is growing very fast. See text for further details.

Example 1.2 Kleiber's law

Many natural laws in the sciences take the form of regression models. For example, after collecting a considerable amount of data comparing the body mass versus metabolic rate (a measure of at-rest energy expenditure) of a variety of animals, early twentieth-century biologist Max Kleiber found that the log of these two quantities are related *linearly*. This linear relationship can be seen visually by examining the dataset shown in Figure 1.9. Examining a similar dataset, Kleiber found the slope of the regression line to be around $\frac{3}{4}$ or, in other words, that metabolic rate $\propto$ mass$^{\frac{3}{4}}$.

This sublinear relationship means that compared to smaller-bodied species (like birds), larger-bodied species (like walruses) have lower metabolic rates, which is consistent with having lower heart rates and larger life spans (we revisit this problem later in Exercise 5.2).

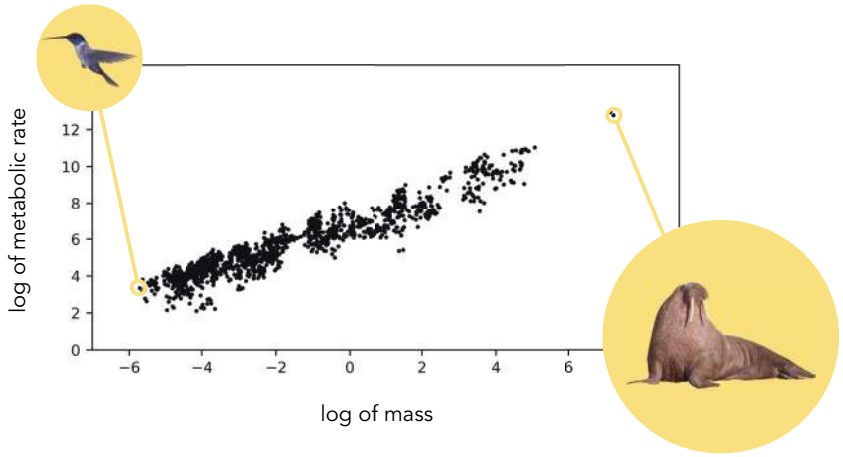

Figure 1.9 Figure associated with Example 1.2. A large set of body mass versus metabolic rate data points, transformed by taking the log of each value, for various animals over a wide range of different masses. See text for further details.

Example 1.3 Predicting box office success

In 1983 the Academy award winning screenwriter William Goldman coined the phrase "nobody knows anything" in his book *Adventures in the Screen Trade*, referring to his belief that at the time it was impossible for anyone to predict the success or failure of Hollywood movies. While this may be true – in the era of the internet – by leveraging data like the quantity of internet searches for a movie's trailer, as well as the amount of discussion about a movie on social networks (see, e.g., [3, 4]), machine learning can accurately predict opening box office revenue for certain films. Sales forecasting for a range of products/services including box office sales is often performed using regression since the output to be predicted is continuous (enough) in nature.

Example 1.4 Business and industrial applications
Examples of regression are plentiful in business and industry. For instance, using regression to accurately predict the *price* of consumer goods (from electronics, to automobiles, to houses) are hugely valuable enterprises unto themselves (explored further in Example 5.5). Regression is also commonly used in industrial applications to better understand a given system, e.g., how the configuration of an automobile affects its performance (see Example 5.6), so that such processes can be optimized.

Classification

The machine learning task of classification is similar in principle to that of regression, with the key difference between the two being that instead of predicting a continuous-valued output, with classification the output we aim at predicting takes on *discrete values* or *classes*. Classification problems arise in a host of forms. For example, object recognition where different objects from a set of images are distinguished from one another (e.g., handwritten digits for the automatic sorting of mail or street signs for driver-assisted and self-driving cars) is a very popular classification problem. The toy problem of distinguishing cats from dogs discussed in Section 1.2 falls into this bucket as well. Other common classification problems include speech recognition (recognizing different spoken words for voice recognition systems), determining the general sentiment of a social network like Twitter towards a particular product or service, as well as determining what kind of hand gesture someone is making from a finite set of possibilities (for use, for instance, in controlling a computer without a mouse).

Geometrically speaking, a common way of viewing the task of classification in two dimensions is one of finding a separating line (or, more generally, a separating curve) that accurately separates two kinds of data.[1] This is precisely the perspective on classification we took in describing the toy example in Section 1.2, where we used a line to separate (features extracted from) images of cats and dogs. New data from a validation set are then automatically classified by simply determining which side of the line the data lies on. Figure 1.10 illustrates the concept of a linear model or classifier used for performing classification on a two-dimensional toy dataset.

Many classification problems (e.g., handwritten digit recognition, discussed below) have naturally more than two classes. After describing linear two-class classification in Chapter 6 we detail linear multi-class classification in Chapter 7. The nonlinear extension of both problems is then described starting in Chapter 10 and throughout Chapters 11–14. Below we briefly describe several further examples of classification to help solidify this concept.

[1] In higher dimensions we likewise aim at determining a separating linear hyperplane (or, more generally, a nonlinear manifold).