Chapter 1

Introduction

Machine learning is a set of approaches that can detect patterns in the data. Types of machine learning include predictive and descriptive reinforcement learning. Two major classes of predictive learning are classification and regression. Examples of unsupervised learning approaches include principal component analysis, clustering, and dimensionality reduction. We can formalize predictive and descriptive learning as density estimation, where we develop probabilistic formulations of the form $p(y|x_i, \mathcal{D})$ for predictive, and formulations of the form $p(y|\mathcal{D})$ for unsupervised learning. Resulting probabilistic models $p(y|\theta)$ or $p(y|x_i; \theta)$ may include fixed number of parameters, or their number may vary according to the size of the training data. Interesting machine learning concepts include the curse of dimensionality, inductive bias, overfitting, model selection, and the absence of a universally best model that would fit all kinds of problem domains. [a]

[a] These lecture notes follow Chapter 1 from **2012-Murphy**. Recommended additional reading is Chapter 2 from **ESL**.

1.1 Purpose of Machine Learning

Machine learning is about learning models from data. More abstractly, given the training data $\mathcal{D}$, we would like to use the data to infer probability distributions. In other words, we would like to build models of the process $p(y)$ that generated the data.

The general task of learning $p(y|\mathcal{D})$, that is, inferring the conditional distribution of variables that define the processes given the data, is very complex. In practice, we are, in most cases, not even interested in this general task. Instead, we are interested only in certain aspects of the distribution, and for these, apply specific types of machine learning, like classification, regression, or clustering.

In terms of applications, machine learning is a branch of artificial intelligence that provides algorithms that can automatically learn from experience without being explicitly programmed. While we will focus on theoretical aspects of machine learning, the reader of this text should place these in practical contexts and consider the tasks such as data acquisition, data cleaning, feature engineering, data cleaning and preprocessing, data visualization, scoring, and estimating the quality and utility of the developed models, and finally, their inclusion within working software and decision support systems. While practically of utmost importance, these engineering aspects will not be at the focus of this course.

1.2 Types of Machine Learning

Supervised Learning

Often, we are only interested in how a subset of variables is generated, while the remaining variables are used to explain the behavior of the variables of interest: $\{y_i, \boldsymbol{x}_i\}_{i=1}^n$. This is *supervised learning*, also known as *predictive learning* or predictions. Here, y is referred to as *response variable*, and $\boldsymbol{x}$ represents a vector of *features*. Depending on a branch of science that deals with machine learning, the dependent variable may also be referred to as a target variable, dependent variable (statistics), or label or class variable (machine learning). The independent variables are often referred to as covariates, independent variables, and predictors (statistics), or features and attributes (machine learning). Supervised learning starts with the training data, which includes n pairs of instantiations of independent and dependent variables. The goal is to learn about $p(y|\boldsymbol{x})$ so that we can make predictions for future or unobserved values of independent variable y for any combination of dependent variables $\boldsymbol{x}$ (see Table 1.1).

When y is a nominal variable, we refer to this type of supervised learning as *classification*. When the nominal variable is two-valued, we deal with *binary classification*, and when the domain of the nominal variable includes three or more values, we refer to the problem as *multiclass classification*. When y is continuous, we refer to the problem as *regression*. Less common cases consider a count or ordinal dependent variable, where we refer to the suitable approaches as *count regression* and *ordinal regression*, respectively. In most cases, the dependent variable y will be a scalar, and we will refer to such cases as *univariate* classification or regression. When y is a vector, we will refer to the problem as *multivariate* classification or regression.

Note that while various machine learning approaches specialize in a particular case, it is often easy to generalize a specific approach to deal with other types of dependent variables. For instance, it is not difficult to adapt classification trees to the regression problems or extend this approach to address multivariate learning.

Table 1.1: A small sample from the famous Iris data set, where Iris flowers are described with four numerical features and are labeled with Iris species. A possible task for this data set is supervised learning, with aim to build a model that predicts species from leaf morphology.

sepal length	sepal width	petal length	petal width	iris
4.4	2.9	1.4	0.2	Iris-setosa
4.8	3.4	1.6	0.2	Iris-setosa
5.4	3.9	1.3	0.4	Iris-setosa
5.2	2.7	3.9	1.4	Iris-versicolor
6.0	2.2	4.0	1.0	Iris-versicolor

Unsupervised learning

We are using *unsupervised learning* when our problem does not include any response variable, that is, when the observations are not labeled. The goal of such learning is again to understand the data generative process $p(y)$, or at least to understand part of its structure.

A common approach to understanding the distribution $p(y)$ is to explain it with a smaller number of factors θ, that is, to learn $p(y|\theta)$, effectively projecting the data into a lower-dimensional space. We refer to such procedure as *dimensionality reduction*. An extreme example of dimensionality reduction is *clustering*, when we try to explain $p(y)$ with a single nominal factor (see Fig. 1.1). In essence, we are trying to group, or cluster, observations.

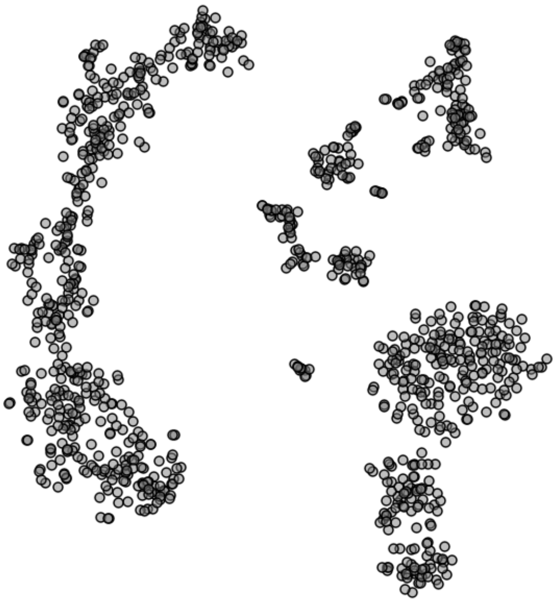

Figure 1.1: A two-dimensional visualisation of blood cells, originally described with expressions of thousands of features. The visualisation was constructed using t-SNE dimensionality reduction, and exposes potential clusters that need to be further analyzed.

Reinforcement learning

Reinforcement learning is about learning actions of software agents in an environment where the goal is to maximize reward. An example of reinforcement learning is to learn the actions of a robot that travels through the maze and receives sensor input. A reward, in this case, could be time spent in a maze. Reinforcement learning is different from supervised learning in not requiring labeled input. Instead, reinforcement learning aims to find the balance between the exploration of uncharted territory and the exploitation of current knowledge. While we will focus on unsupervised and supervised learning in this course, we will only dive into reinforcement learning in one of our final sessions.

1.3 Models and Learning

A *model* $\mathcal{H}$ is in the most abstract sense a collection of distributions (densities, functions, ...). The elements of a model depend on our task.

> **Example: Simple linear regression - statistical model**
> The simple linear regression is a set of densities
> $\mathcal{H} = \left\{p(y|\boldsymbol{x}, \boldsymbol{\beta}, \alpha, \sigma) = dnorm(\boldsymbol{\beta x} + \alpha, \sigma^2), \boldsymbol{\beta}, \alpha \in \mathbb{R}, \sigma > 0\right\}$

> **Example: Simple linear regression - function approximation**
> The simple linear regression is a set of functions $\mathcal{H} = \{f(\boldsymbol{x}, \boldsymbol{\beta}, \alpha) = \boldsymbol{\beta x} + \alpha, \boldsymbol{\beta}, \alpha \in \mathbb{R}\}$

It is important to reinforce the view of a model as a set of hypotheses. Learning is the process of expressing a preference for certain hypotheses based on evidence (data). Choosing a particular machine learning algorithm, or in other words, choosing a particular model means expressing a preference for a certain type of hypothesis. The logistic regression model, in its basic form, will construct a model which will linearly separate the parameter space of the data instances to, prefarably, separate the data instances from either of the two classes. Separation plane constructed by classification trees may be much more complex and, implicitly, require many more parameters for its descriptions. The choice of the model also entails the choice of the complexity of the hypothesis, which in turn is related to the goodness of fit, overfitting, explainability, and other issues we expose in the text below.

A model is often also referred to as a hypothesis or set of hypotheses. Learning is often referred to as training the model, fitting the model/parameters, estimation.

Learning is the process of selecting elements of $\mathcal{H}$ based on some utility and using data. This is general. In practice, we can select a single element (a single density, function, distribution; as in the two function approximation examples above), a set of elements or even weight each element, for example, a distribution across all elements, as in Bayesian approaches.

Learning is in most cases just a problem in *computation* to be addressed through mathematical, numerical, algorithmic procedures. For parametric models, we typically do *least-*

squares or *maximum likelihood* estimations to obtain *point estimates* of the parameters of the model. That is, by learning, we select a single model. Learning thus becomes an *optimization* problem, or *Bayesian inference*, which is an *integration* or optimization problem, if we do some sort of *structural approximation* or *MAP*.

Parametric and Nonparametric Models

If the set $\mathcal{H}$ can be parametrized with a finite number of parameters, we call the model *parametric*. Otherwise, it is *non-parametric*. A parametric model captures all information about the properties of the data within its fixed number of parameters. Predictions using non-parametric models require knowledge about the current state of the system, that is, require access to the current data.

> **Example: 1-nearest neighbor model - a nonparametric model**
> $\mathcal{H}$ = { all functions f that can be expressed with a set of points (data instances) and the rule that $f(x) = y_i$ of point x nearest to x_i, according to a chosen distance metric} (see Fig. 1.3)

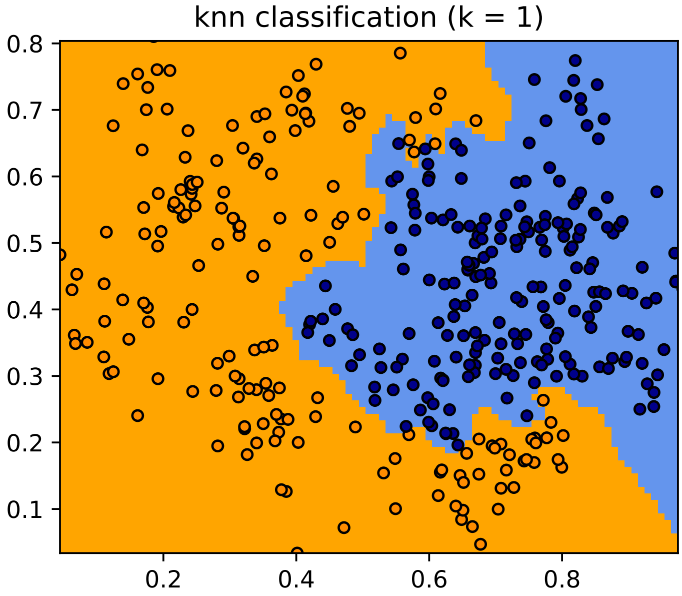

Figure 1.2: Decision boundary of a 1-nearest neighbor model trained on a two-featured binary classification data set with 380 data instances (170 in one class and 210 in the other) as shown on a figure. The decision boundary is complex and may substantially change with any addition on removal of the data.

Parametric models are easier to compute than non-parametric models. Non-parametric models are often more complex and grow with data. Parametric models depend on the data only through a finite number of parameters, while in non-parametric models, the complexity of the model depends on the training set. Researchers mostly prefer parametric models because it may be easier to estimate its parameters, perform predictions, and tell a story about the data according to a parametric model (e.g., sensitivity analysis, effects of the changes in parameters, parameter interactions). In this sense, parametric models are more prone to interpretation by domain experts. In parametric models, the parameter estimates may have better statistical properties compared to those of non-parametric regression.

Parametric models make stronger assumptions about the data; the learning may be successful if these assumptions are valid, but the inferred predictors may fail if these assumptions are violated. Think of modeling a sine curve with a linear regression model. A non-parametric algorithm is computationally slower but makes fewer assumptions about the data. In the (overly) simplified view, the tradeoffs between parametric and non-parametric algorithms are in computational cost and accuracy.

Notice that non-parametric models are related to *lazy learning*. Lazy learning methods generalize the training data at the time of prediction. This type of learning is an alternative to *eager learning*, where the system tries to generalize the training data before receiving queries. An example of the lazy learner is a K-nearest neighbor algorithm. Lazy learners may have an advantage in real-time environments, where the training data changes in time and models trained in the past become obsolete in a relatively short time due to emergent new data and changes of the distributions and underlying processes that generated the data.

1.4 Challenges of Applied Machine Learning

Model Evaluation and Selection

In theory, assuming uniform distribution over all possible datasets, there is no single best model. In fact, and again, in theory, no model is strictly better than any other model. This is in the literature referred to as *no free lunch theorem*, which states that any two optimization algorithms are equivalent when their performance is averaged across all possible problems (**2005-Wolpert-Macready**).

In practice, however, some characteristics are more common in datasets, so some models and algorithms perform better on average because their assumptions (*inductive bias*) better match the characteristics of the data generating process.

The above makes *model selection* a key part of applied machine learning. In order to train a model, we should define some measure of utility we would like to optimize. A trivial approach is then to select the model with the best utility on our available data. However, estimating the model's utility on the data it was trained on is biased and optimistic. In practice, the model's utility on the training data (so-called *in-sample* error) may be substantially better than on independent test data (*out-of-sample* error or *generalization error*). This effect is also known as *overfitting*, and it is something that we want to both detect and prevent.

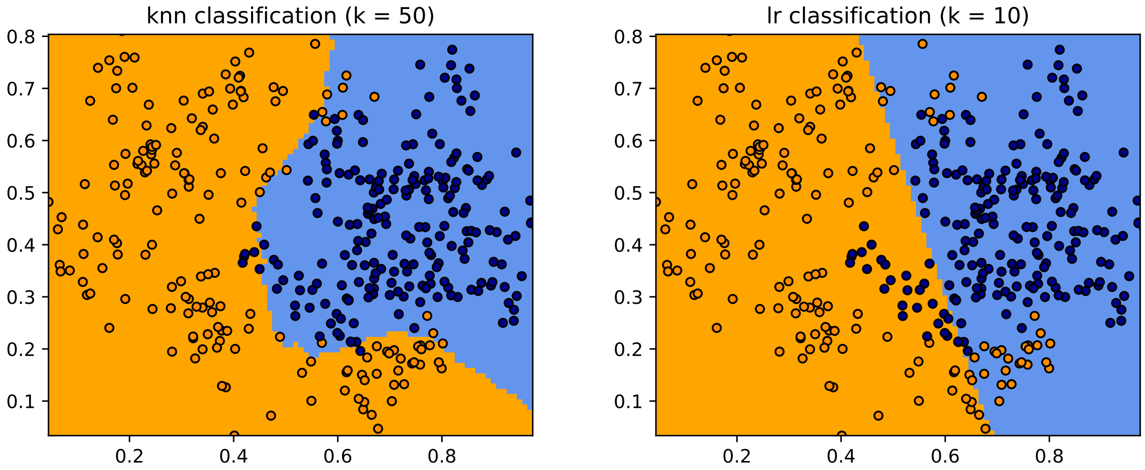

Figure 1.3: Decision boundary on a two-featured binary classification data set as inferred by the nearest neighbor algorithm with K=50 and by logistic regression. Which model would perform better on new data?

Overfitting

Overfitting occurs when a machine learning model trained on a (limited) data set captures the data's noise instead of the underlying data generation processes. This modeling error occurs when an inferred hypothesis is too closely fit a limited set of data points or when a model is too complex for a given data (e.g., Fig. 1.4).

The related practical challenges include knowing when overfitting occurs and finding the right remedy for overfitting. Another challenge is to avoid modeling procedures that led to overfitting. None of these challenges is trivial, and beginners or even quite experienced practitioners often make mistakes that lead to overfitting and consequentially report over-optimistic scores for their modeling procedures. For instance, it has been found that some (if not most) of significant reports on the analysis of microarray gene expression data sets at the break of the century included overfitting (**2003-Simon**). Common reported mistakes included feature selection before cross-validation or class label-informed feature selection before data visualization. Reports on good accuracies are, despite teachings in data science, present also in recent literature, as reported by **2019-Vandewiele**. The authors examined reports on the analysis of a collection of electrohysterogram signals. There, related reports oversampled the data before cross-validation and hence falsely obtained almost perfect accuracies.

Model Complexity and Effective Number of Parameters

More complex models are more likely to overfit (see Figs. 1.6 and 1.7), but the "right" complexity of the model may depend on the amount of data we have (see Fig. 1.8). In parametric models, especially linear models, the model's complexity easily be measured in terms of

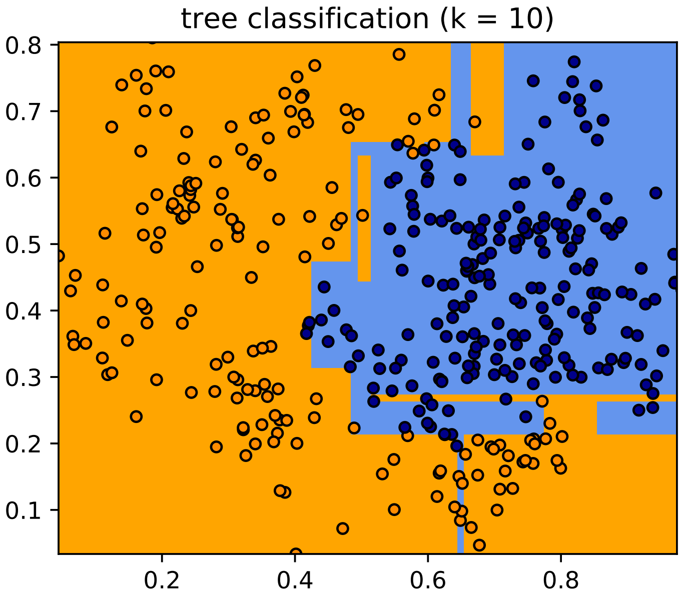

Figure 1.4: Classification trees would often overfit the training data. Figure shows decision boundary of a tree where the allowed maximum tree depth was 10. The decision boundary is complex and often covers single-case exceptions.

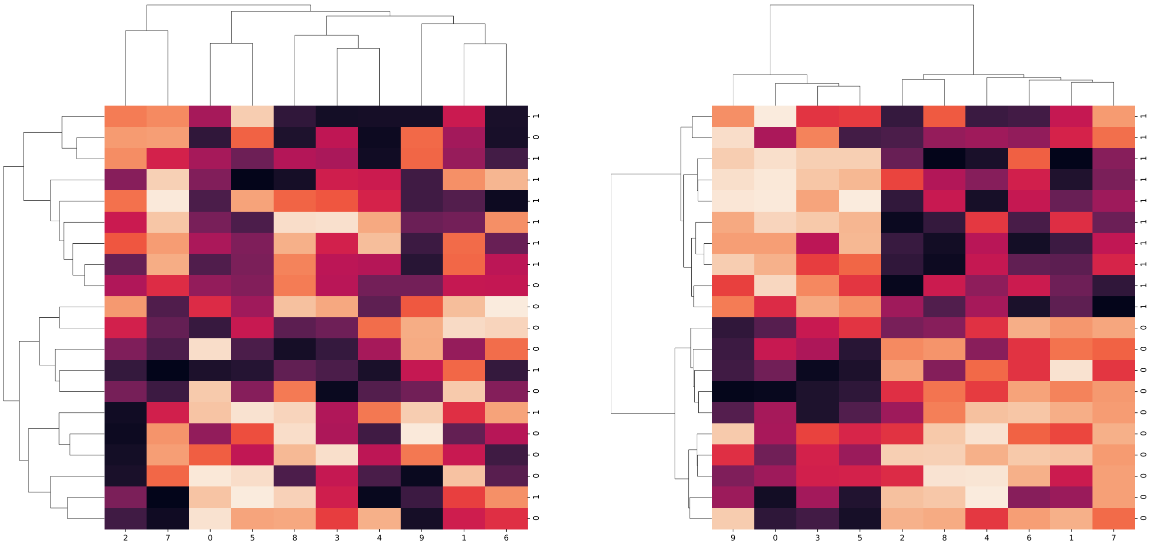

Figure 1.5: Co-clustering of a random data set with 20 instances and 10 features (left), and co-clustering of a similar data set with 10000 features, of which 10 features were selected that best correlate with a binary label (right). Notice a clear pattern of colors and shades in the right heatmap, which should not be there if correct data preprocessing procedures were applied.

the number of parameters (or degrees of freedom). For non-parametric models, the theory is more complex (*e.g.*, Vapnik–Chervonenkis dimension) and introduces the concept of the *effective number of parameters*. Typically, non-parametric models have a higher number of effective parameters and are thus able to better fit the data but also more prone to overfitting. But they are more difficult to interpret.

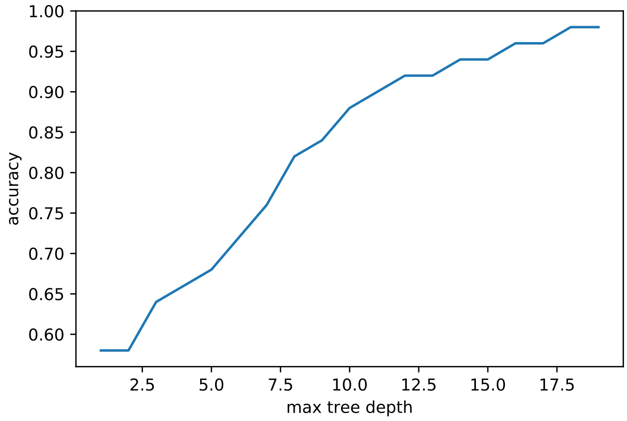

Figure 1.6: A classification tree accuracy on a random class-balanced binary classification data set with one feature and 50 data instances. Trees were grown to a specified maximal depth. More complex trees better fit the training data.

Practical Utility of Machine Learning Models

In practical applications, there are other dimensions (other than predictive accuracy, etc.) that we need to consider:

- *Computational aspects* include runtime complexity and resource consumption and are specifically relevant when modeling large data sets and streaming data, where models need to be adapted frequently and where there is inherent concept drift. Notice that while some computational can be mitigated with modern hardware, we must understand that even considering pairwise feature interactions requires computation squared in the number of features, not even counting the number of data points. All alternatives are based on discarding some information: data subsampling (sublinear learning algorithms), feature selection, or discarding higher-order feature interactions.

- *Implementation aspects*, where data scientists need to decide which parts of the analysis procedures to implement on their own, gaining in flexibility, and for which to rely on already existing implementations. These later may also be limited in terms of data type (e.g., sparse or full), scalability (multi-core, multi-processor, or multi-GPU computing), and data access (e.g., Excel tables, SQL databases, or data in the cloud).

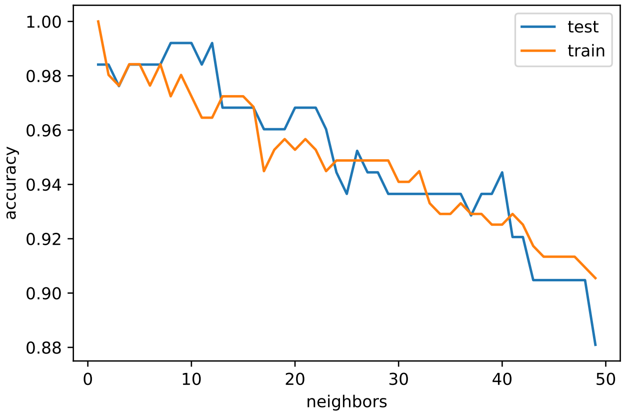

Figure 1.7: A training and the test-set tradeoff for k-nearest neighbor model. On the training data, the accuracy falls with raising k, while on the test data set, the accuracy peaks at around $k = 10$. Hyper-parameter estimation is one of the key issues when selecting the most appropriate model.

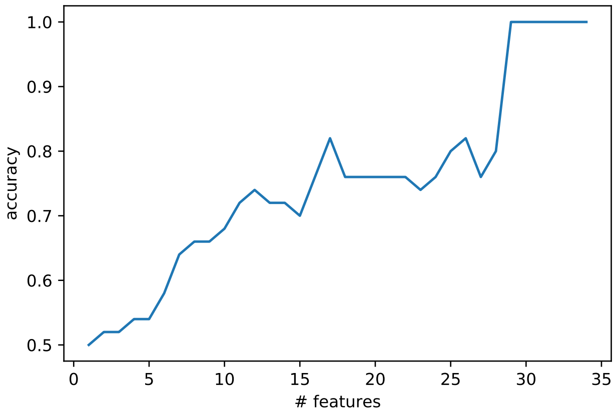

Figure 1.8: Logistic regression is more robust to overfitting than classification trees but succumbs as well when given a sufficient number of features. A graph shows a training error on a random 50-instance binary classification data set when adding up to 35 features.

- *Interpretability*, which often refers to the question if the model is readable or can be converted to a readable format. And if it is readable, is its interpretation easy (e.g., just a few if-then-rules) or impossible (e.g., a long list of rules or a large classification tree).
- *Explainability*, often confused with interpretability, places a model within a context of a problem domain and asks a question did we gain any new knowledge. To achieve explainability, one would often need to combine the interpretation of the model with extra formalized knowledge about the domain (*e.g.*, feature groups, ontologies, rules, and similar).

Every modeling paradigm we introduce in this course should and will be discussed from these perspectives. Notice that most data science courses often focus on predictive accuracy alone; the intended audience may often forget that other issues are equally or even more important in practice.

Curse of Dimensionality

In practice, the complexity of the models we want to fit is not bound only by computational resources but also by the fact that a linear increase in the number of variables can increase exponential increases in the number of possible configurations. Therefore, the amount of data that would be required to distinguish between these configurations is impractical.

The curse of dimensionality may also inhibit or even cripple some machine learning methods. For instance, k-nearest neighbors may work well on two-dimensional data, but as soon as the number of dimensions increases, to a few more dimensions, the algorithm fails. To illustrate this point, consider embedding a small d-dimensional cube of side s inside a larger unit cube (Fig. 1.9). Let the data be uniformly distributed within the unit cube. Suppose we estimate the density of a class labels around a test point $\boldsymbol{x}$ by growing a smaller hyper-cue until it contains a desired fraction f of the data points. The expected length of this cube will be $s(f,d) = f^{1/d}$. Say, with $d = 10$ and to base our estimate on 10% of the data, the length of the smaller cube would need to be $s = 0.8$. The approach, despite the name “nearest neighbor” is no longer very local, as even with the modest feature sizes, it relies on data points that are far away. Even with 1% coverage, the size of the small cube needs to be substantial, as $s(0.01,10) = 0.63$. With a number of features growing, we quickly have to start taking into account points that are not close or risk increasing variance.

References

Freund, Y. and R.E. Shapire (1997). “A decision-theoretic generalization of on-line learning and an application to boosting”. In: *Journal of Computer and System Sciences* 55, pp. 119–139.

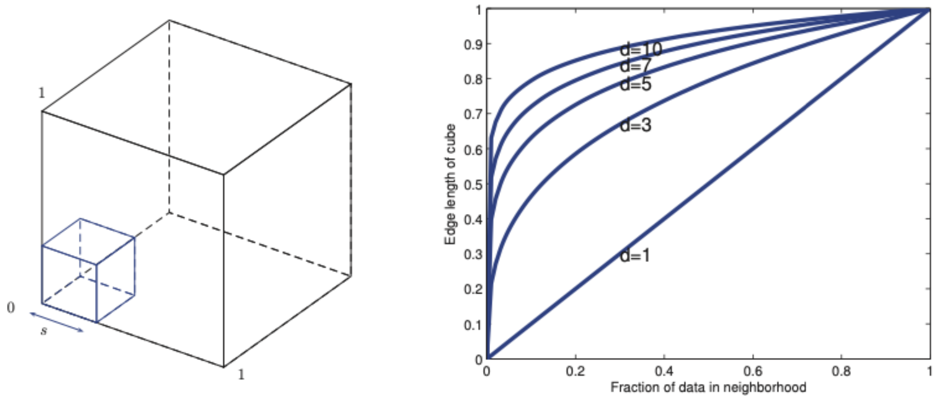

Figure 1.9: We embed a small cube within a unit cube (left) and assess the length of the edge of a small cube to cover a fraction of uniformly spread data. Graphs borrowed from **2012-Murphy**.