

Chapter 2

Fundamental Concepts

Abstract Data mining aims to go beyond the mere describing of a set of data to identify those relationships – patterns – that exist between seemingly unrelated items. In this chapter, we pose some of the types of questions we might need to answer, basic concepts such as that of overfitting, bias, normalization, and dimensionality. We also introduce a broad, but certainly not exhaustive, range of techniques to illustrate how widely we can cast our net when looking for techniques to help us. For example, principal component analysis (PCA) can help us with the curse of dimensionality. We also list a number of statistical measures that we use consistently in data mining. In fact, statistical analysis is still one of, if not the, major sets of techniques used in analysis. We also include some important topics such as distance; distance is used in many techniques as a measure of similarity.

2.1 Introduction

Consider the following questions:

- ?. Is there a correlation between drug X and condition Y?
- ?. An increase in Z is seen when conditions A and B occur. Do the occurrences of these two conditions cause the increase in Z or is it the other way around?
- ?. Is there a correlation between highly expressed genes and highly expressed proteins?
- ?. Given a set of microarray sample results, can we accurately diagnose the disease? Can we predict what the outcome will be for a given therapeutic treatment? Can we recommend the best therapeutic treatment?
- ?. Why are we seeing that as X changes, Y changes also?

Questions like this need to be considered more and more often as we extend our knowledge, expertise, and the data we capture through our business and scientific efforts. As we better understand the genome, proteome, and transcriptome, the nature of these questions becomes more complex and may not always be able to be

predetermined. Or the nature of the question may be able to be predetermined, but may arise from patterns within the data.

As databases grow in size, the amount of data describing any given facet of data stored in the database becomes a more accurate reflection of the distribution of the data itself. For example, the more individuals for whom a salary is recorded will typically give a more accurate reflection of the statistics associated with that data point. What can be as interesting as the statistical analysis on individual pieces of data is how data interrelates to other data and how we can elicit new information from the data we are capturing.

As we analyze and model processes which occur around us, we find a surprisingly complex set of interactions between data characteristics. The process of human in vitro fertilization, for example, uses over 60 different characteristics to determine which embryos should be implanted (Witten and Frank 2005).

In 1985, a personal computer with a 10-Mb hard disk was uncommon. In 2005, a laptop can be purchased with 160 GB of hard disk storage. Inexpensive storage, complex analysis software, and automation have enabled us to capture data more broadly and deeply than ever before. Each year, organizations significantly increase their data storage volumes – not decrease. In an informal straw poll of IT professionals, the authors could not find a single person that confirmed their company had reduced its data storage volumes over the previous year! To put this in perspective, the average novel contains 60,000 words. If each word is assumed to be 10 characters in length, this is 600,000 characters. If a character can be stored in a single byte of storage, this equates to 600,000 bytes, or 586 Kb (or 0.00056 GB). Thus a 106 GB disk can store over 286,000 novels! Large databases run into the upper gigabyte range, and many are now entering the terabyte range.

In the life sciences, we are typically not dealing with novels, but instead with large amounts of numeric and string data. It is not unusual for us to deal with sequence data consisting of thousands of base pairs. We also encounter circumstances where the data is contained in several disparate databases, disparate in location and content. These individual sources, usually containing complex and heterogeneous data, must then be brought together and managed as a coherent whole. This process is easier said than done, as we shall describe later.

But there are many processes for which deterministic, algorithmic approaches are not yet available. Weather prediction, fraud detection, and protein conformation are three widely studied. Yet for each of these subject areas, significant amounts of data are captured daily.

Today, it is fair to say that data overwhelms us. A 30-s Google session returned 105,000,000 hits for the keywords “data mining,” 7,580,000 for the keywords “data mining machine learning,” and 37,500 for the keywords “data mining machine learning-protein conformation”! While informal, this illustrates that we are generating data at a much faster rate than we understand the data.

In data mining, we are interested in discovering patterns that are hidden in data. In order to do this, we use techniques from machine learning, statistics and inference, and data analysis, to name a few. In order to accomplish this, we use these techniques with automated searching techniques to identify patterns which can then be validated and then used for prediction.

We can, therefore, also consider data mining as being about problem solving through analyzing existing data by finding patterns within the data. Patterns can be expressed in any number of different ways. On one end of the spectrum, we have a completely opaque pattern – the black box – while on the other end of the spectrum is the completely transparent pattern that allows us to see and understand its structure. Witten and Frank (2005) call patterns which are represented in terms of structures that can be examined, reasoned about, and used for future decision-making *structural* since they capture the decision structure in an explicit way; that is, they help to explain the data in some respect. We will use the same definition since the objective is to uncover and describe such structural patterns in data.

We often find that certain patterns occur frequently enough in our data for us to consider them specifically. Inferring that these patterns will occur with a similar frequency in data yet unseen offers valuable insight into the larger dataset and to other patterns we can uncover. One correlation of this is that we may be able to identify not only patterns but sequences and other structures within the data. Using a consumer-based example, buying a camera could lead to buying additional memory cards, an intuitive sequence, but how often does this also lead to buying photo editing software and even a new PC?

Data mining techniques are often compartmentalized into statistical and machine-learning categories. However, many techniques use a combination of techniques from both categories, either sequentially or in parallel. Further, many problems can be solved through either type of approach. An example of this can be seen in the area of classification and regression trees.

In order to identify patterns in data and to use the results for decision-making in the future, we have to have methods which *learn*. On one level, learning can be thought of as searching through some space of possible definitions to determine (the) one that best fits the data we have. Theoretically, we can consider this as a problem whereby we enumerate each definition in our space and eliminate those which do not fit the *training data* (examples) we are given.¹ With each example in our training set, the set of definitions remaining will reduce or remain the same until we have exhausted our training set.

We can then be left with a set of definitions, or in the extreme, a single definition. In the latter case, it is the target we are looking for. In the former case, the set of definitions left may still be used to classify previously unseen definitions. If a previously unseen object matched every definition in our set, we can definitely consider it to have matched the target. If it does not match any definition in our set, we can definitely consider it to be outside of our target. However, the ambiguity and complexity comes in when some subsets of our definitions are matched. We have to be careful since if we classify the object into some classification, the definitions in our set for which the match failed will be eliminated, which may not be acceptable.

¹ Note that we can elect to exclude those that *do* match our examples, or those which *do not* match our examples.

This overclassification would make the system fragile and error-prone as it can overfits the training data when incremented with the previously unseen object.

Continuing with our conceptualization of learning as search, the first question that needs to be addressed is how the search space is defined or enumerated.

Consider that each *characteristic* or attribute in each concept definition can have a range of values. As an example, consider the following set of characteristics that we might define as our *patient* object:

- Age (0–99)
- Gender (male, female)
- Race (Caucasian, Asian, African, Hispanic)
- Pregnant (true, false)

If we were to only consider the following attribute set, we would have a total of $100 \times 2 \times 4 = 800$ combinations we would have to consider *for each rule*. In reality, we will have vastly more than this – both in terms of the domain of values for each characteristic and the number of characteristics themselves. Each of these combinations could be implemented as a *rule* such as

```
if age > 16 and gender = female
    and pregnant = true then inclusion = false
```

This results in a huge number of possible rules.

We could decide to create a rule set with a limited set of rules. For example, if we have 20 training examples, we might elect to generate a rule set containing an equivalent number of rules. In this case, we have a total of 800^{20} or 1.15×10^{58} possible rule sets! As we stated, the type and number of attributes we used to come up with this number are patently naïve and would not provide a rich enough environment for real-world problems.

This *enumeration* issue is definitely a challenge, although various techniques are available which can help. But it is not the only issue. A related issue involves the fact that in the majority of cases, a unique object is rarely returned: the process will not typically converge on a unique result.

There are only three possible outcomes for processing a search space: either a unique result is returned, as discussed above, or no result is returned, or more than one result is returned.

When no result is returned, it means that the language we are using to describe the problem under study is not rich enough to capture the concept, or that there is too much noise in the set of training examples. An example may be incorrectly classified because the values provided contain some level of noise. If we were to consider a classical problem in artificial intelligence, that of handwriting recognition, for example, then the scanned character images may contain noise due to dust on the scanner platen. This could result in a “c” character being mistakenly classified as an “o” character if the dust was in the right position.

When more than one result is returned, it is typically because the example set we have used to train our system is not sufficiently rich to allow us to obtain just the

single result we are ultimately striving to obtain. In such cases, the goal may be to return the “best” result from the set of results, and we may use some additional criteria for this purpose.

We can consider this *generalization as search* problem from another perspective also: as hill climbing in the problem space to find the result which best matches the example set we use for training our system, according to some specification of matching criteria. This is how most practical machine-learning methods work. It should be noted, however, that for any real-world problem, it is infeasible to exhaustively search the problem space; most algorithms use heuristic search methods. These, obviously, cannot guarantee to find the optimal solution.

In this chapter, we introduce a large number of different techniques which can be used within the data mining effort. Prior to this, we make note that data mining and knowledge discovery use different definitions to mean the same thing. We particularly like the terms used in Witten and Frank (2005):

- *Concept description* refers to the thing to be learned.
- The input takes the form of *concepts*, *instances*, and *attributes*, where an instance is an independent example of the concept and comprises a set of attributes.

We also pause to introduce the concept of *supervised* learning and its complement *unsupervised* learning. We will define these more precisely and formally later, but at this stage it suffices to say that with supervised learning methods we are provided with output *labels* for each training set instance; with unsupervised learning, the method needs to identify such labels from the training data itself. Both of these areas of machine learning provide significant value in their respective domains.

2.2 Data Mining Activities

Every person initiating a data mining activity will do so with a particular form of analysis that he/she would like to perform. This objective will typically be described as a *data mining query* that will be used as the primary input to the system.² We do, however, need to decompose this query a little further in order to direct the mining effort itself to optimize the overall process as much as possible. While this decomposition typically occurs intuitively, we dissect it here in order to provide context for some of the discussions later.

- Mining data subset

In any given mining activity, only a subset of the overall data repository will be relevant and of interest to the user. This will include all the dimensions of the data warehouse, subset of attributes, and subset of the data domain itself.

² Note that we consider the data repository to be an integral part of the data mining system, thus the data itself – training data or otherwise – is not designated as an input.

- Mining functions

This defines the type of function that is to be performed within the mining activity. Examples include classification, outlier analysis, association, or any combination of such functions.

- Contextual knowledge

This is the knowledge and information available to the user or subject matter expert that relates to the domain under study and provides significant value in guiding the overall discovery process as well as for pattern evaluation once such patterns have been uncovered. We shall consider several techniques, such as concept hierarchies that provide support in this area.

- Value measures

These are the measures, thresholds, and probabilities which we will primarily use to evaluate the patterns uncovered by the mining effort. However, they may also be valuable in helping to direct the mining process as well as postdiscovery.

- Visualization representation

How are we going to visualize the output from our efforts? Tables, charts, cubes, graphs, rules, decision trees, and many other representations may be used.³

As we shall see, these concepts and components come together in the models and query languages we use to physically perform the mining activity.

2.3 Models

In much of our discussions on data mining, the concept of a *model* is used. A model is a representation of reality, but usually in a constrained domain. While it encapsulates the essential characteristics of the part of reality it is representing, any aspect of its real-world counterpart which is extraneous to what is being studied is usually omitted. An example of this is the primary structure of a protein, a string of amino-acid representations that provides us with significant scientific value, but which does not encapsulate all of the characteristics of a protein and is clearly not the protein itself. Thus, a model is a theory or hypothesis about some aspect of reality.

We often need to consider multiple hypotheses rather than any single one since the data we are dealing with may be effectively explained by more than one model. We can then use the multiple models to direct our efforts and seek additional data to distinguish between the models under consideration. Using this additional data, we can iteratively evaluate the plausibility of each model in light of the new data we are using, with the goal being to identify the model which most effectively satisfies the data. We shall return to this concept throughout our discussions.

Developing models, therefore, provides a mechanism for us to define a facet of reality that we are interested in and to overlay this with our data repository. Once

³The authors particularly like the creative approaches to displaying information stimulated by Tufte (1990 #531, 1997 #538, 2001 #535).

we develop a model, we will often go through an iterative process of training the model to perform efficiently and effectively against our data, but use a smaller set – the *training data* – that will allow us to tune the model’s architecture and parameters. Since the overall purpose of our model is to generalize to finding patterns in data as yet unseen, it is important to ensure that the model we have developed is not so specific to our training data that its performance is severely impacted as the data volume, and distribution, grows.

2.4 Input Representation

Since every data mining effort works on a set of input records, rows, or *instances*, and this set of inputs is what we will classify, cluster, associate, or otherwise process, representing the data effectively is an important part of the overall architecture.

The most common, and arguably pragmatic, way of representing the input data instances is for each such input to be comprised of a predetermined set of attributes. While this is undeniably valuable, when relationships between instances or attributes are our objective, this representation may not be the most effective.

Consider the example where we are interested in drugs which are *agonists*, a compound which binds to a receptor and produces a biological response. If we have 20 compounds in our input, along with 20 receptors, say, then this would result in $20 \times 20 = 400$ pairs, with the likelihood that most of the pairs do not have an agonistic relationship. Further, if we are interested in whether any of our compounds are *coagonists*, we would similarly have 400 pairs where most of the pairs of compounds are not coagonists. Specifying a “yes” or “no” value for each combination of compounds will result in a large number of rows in which the value is “no,” as shown below:

Compound 1	Compound 2	Coagonist?
C1	C2	Yes
C1	C3	No
C1	C4	No
...	...	
C2	C1	Yes
C2	C3	Yes
C2	C4	No
...	...	
C3	C1	No
C3	C2	Yes
C3	C4	No
...	...	
C4	C1	No
C4	C2	No
C4	C3	No
...	...	

An alternative approach is called the *closed world assumption* which is based on the presumption that what is not currently known to be true is false. Here, we can just specify the true (“yes”) values and assume that any “missing” values are false (“no”). In our example, the table would be

Compound 1	Compound 2	Coagonist?
C1	C2	Yes
C2	C1	Yes
C2	C3	Yes
C3	C2	Yes

In the above table, any combination not included (e.g., C4 and any of C1, C2, and C3) is assumed to have a value of “no.”

As we shall see later, this example of converting a relationship into an appropriate form for data mining is vital and often as challenging as the mining effort itself. We shall consider concepts such as *denormalization* to allow us to flatten tables for more effective analysis.

The above examples informally introduce the concept of qualitative attributes, which are usually expressed as categories (e.g., sex, zip/postal code), and quantitative data which are intrinsically numerical quantities (e.g., age, temperature, cholesterol level). Qualitative data is *nominal* if it is categorized, but has no implicit or explicit order. For this type of data, the equality or inequality relation is the only operation we can establish. If the data is *ordinal*, there is an ordering of the values, and this allows us to establish an ordering ($>$, $<$, $=$), but not to make any statements on the differences between the categories. For example, we may be interested in academic achievement and assign values for non-high school graduation (0), high school graduation (1), college education (2), bachelor’s degree (3), master’s degree (4), and Ph.D. (5). Here we can clearly see that there is an ordering, but to ask whether “college education – high school graduation” (2–1) is the same as “bachelor’s degree – college education” (3–2) is nonsensical. However, we can intuitively say that “bachelor’s degree $>$ college education” in this scheme.

2.5 Missing Data

Any analytical activity is only as good as the data being analyzed. If the data contains erroneous data, the results will at least be skewed, if not totally worthless. This pretty obvious statement does contain some not so obvious issues that we will discuss in more depth later. One of the most visible aspects to this issue, however, is where data is simply not available to us.

In some respects, missing data can be even more problematic in that we do not know what the missing data should be – the old “would everyone not in the room please raise their hands” syndrome. This problem takes on an additional complexion as we deal with larger and larger numbers of attributes since missing data is not as easily dealt with and it can also cause problems with our models that are not readily apparent.

Our first question should be to ask why the data is missing. Is it because the instruments capturing the data malfunctioned? Is it because the people who are the sources of the data declined to provide the data? Did our experiment change midway through? Did (any) aggregation task eliminate the data? Was it never collected in the first place?

The reason *why* the data is missing is as important as how we will deal with the fact that it does not appear in our dataset. To be able to answer some of the questions we raise in this section, we will need access to people who are subject matter experts on the data itself. While this knowledge may be readily available to you – you may even have that knowledge yourself – it may also reside outside the team. It is important not to minimize this issue because only such subject matter experts can tell us if there is additional significance to the missing data other than the fact that it is missing.

Many of the techniques we cover will allow us to handle missing values, even very simple methods such as the 1R method that we discuss below. However, a conscious decision must be made as to how we wish the various methods to handle missing values.

One method is to explicitly include a value of “missing” for those attributes where a value is unknown. In categorical data, we can include an additional class, say called “missing,” for any value which is not provided. For numeric data, we could use values such as 0 or 1, depending upon what has the least impact.

The question of impact is an important one. If we are going to apply mathematical operators to the data, a value of 1 may have a minimal impact, whereas a value of 0 could result in the dreaded “divide by zero” error. Of course, simple arithmetic activities may require a value of 0 to avoid inflating our counts.

We shall return to this issue as we discuss various methods throughout this book since the approach we take will often be driven as much by the method as by the data itself.

2.6 Does Data Expire?

Food, medicines, and many other items we interact with on a daily basis have a “sell-by” or “use-by” data. Can the same be said for data? Does it expire? Is there a point after which using the data would, or could introduce errors into our model? As is always the case, the answer is “it depends.”

If we are using information to draw conclusions of a transient nature, such as recommending a therapy, the answer is typically yes, the data expires. A patient’s condition will change over time and may change very quickly. We would not want to use results from a blood panel taken 3 months ago, for example.

The above example, however, is probably not going to be where our issues lie. We may need to be very careful if we are using geographical data for individuals, since addresses will often change. In such cases, we need to determine whether or not such changes can affect our model: does it matter if the addresses of patients in

our dataset have changed? The answer will depend on what conclusions we are trying to draw from our models and also what the underlying meaning is for both our input data and our models. Once again, we need to have a good understanding of our data, or at least, access to someone with a good understanding of our data.

2.7 Frequent Patterns

We have already alluded to the fact that frequent patterns, patterns that occur again and again in our data, provide a natural starting point for data mining. After all, we would definitely want to be cognizant of these quotidian patterns before analyzing the more esoteric patterns inherent in our data.

Such patterns provide additional understanding of our data in that sequences and other structures will often be inherent within the frequent patterns themselves. For example, the use of stimulants will obviously show increased heart rates and other heightened metabolic factors. Can we also elicit other subsequent physiological relationships from our data?

Association analysis, described below, provides a good first step in this process. We are interested in the association (or implication) between two or more facets of the data. For example, an MIC⁴ of 4 for a *Streptococcus pneumoniae* isolate taken from a patient when tested against vancomycin and ampicillin would be a good indicator of multidrug resistance for that isolate, a determination we can make without testing against other compounds. We may describe this rule as follows:

Vancomycin MIC > 2 and ampicillin MIC > 2 → multiDrugResistance

Confidence = 75%

DataSupport = 4%

Our measure of confidence, also called accuracy, means that 75% of the time that a vancomycin MIC > 2 occurred and that an ampicillin MIC > 2 occurred, multidrug resistance also occurred. Similarly, our measure of data support, also called coverage, means that 4% of the instances in our dataset showed that isolates that had an MIC value greater than 2 for vancomycin and an MIC value of greater than 2 for ampicillin **and** a designation of multidrug resistance occurred together. There is a subtle nuance to the above values: in one, we are looking at how often these characteristics physically occurred in the data (data support), whereas in the other, we are looking at how often the characteristics to the left of the arrow inferred the characteristics to the right of the arrow (confidence).

⁴ MIC stands for minimum inhibitory concentration and is the minimum concentration of a drug that inhibits the growth of a microorganism after overnight incubation. It is an important value for antimicrobial agents as it is used as an indicator for antimicrobial resistance and antimicrobial agent activity.

2.8 Bias

Our intuitive concept of bias will serve us well as a launching point for considering how and where this issue rears its ugly head in our data mining processes, models, data, and objectivity. In statistics, a *biased sample* is a sample in which the members of the statistical population are not likely to be equally likely chosen. A *biased estimator* is one which consistently overestimates, or underestimates the quantity it is intended to estimate. Either of these conditions could lead to our concluding invalid results. In data mining and machine learning, these and other forms of bias can leak into the system.

As we have defined elsewhere, our data mining models will typically be tested and validated against a subset of the data on which it will operate when fully implemented. *Generalizing* the model is therefore fundamentally important. In one respect, therefore, we can consider generalization as a search through some state space of possible descriptions, in order to identify the one which best fits our data. For example, we may have a set of results from our data mining model expressed as a set of rules. Suppose that we list every possible combination, or set, of rules and we then iterate through them to find the subset which best fits a particular set of data we have. This is certainly a large job, but not impossible.⁵ What we need to be concerned about when viewing generalization in this way is that the order in which the state space is searched, and the way in which the model may be overfitted to the training data must be considered throughout. Even the language we use to define the problem may introduce a level of bias.

We will return to this concept at several points through this book, but introduce several generic forms which must be carefully considered and which are relevant.

The language we use to describe the concepts that we are targeting can, in and of themselves, impose constraints on what we are trying to learn. We need to ensure that the language we use in defining what we are trying to learn and, by extension, what our models tell us does not prevent us from representing every possible subset of examples. While this is not always a practical issue due to the fact that the set of possible examples is usually very large, it can come into play if the language cannot allow for disjunctions (“or”) to be represented⁶: in such a case, our model may not be able to achieve good performance. A more usual form of language bias that we see comes from domain knowledge. A subject matter expert will often know if certain combinations of attribute values can never occur. For example, a microbiologist would know that microorganism A can never be resistant to antimicrobial B and also that microorganism C is not a β -lactamase producer. More generally, this type of situation occurs when one attribute implies another. In such cases, knowing

⁵For more details on generalization as search, the reader is referred to Witten (2005 #11, pp. 30–32).

⁶We purposely exclude discussion of this issue herein as most languages we would encounter in practice will allow conjunctions and disjunctions.

these facts and eliminating such data can significantly reduce the state space our model has to search.

We will typically encounter data mining problems where there are many alternative “solutions” to the problem; we must then determine how we find the “best” one. The definition of “best” will often be specific to the problem at hand, although we often use a criterion such as simplicity. However, under any circumstance, the question of which solution has the best fit to the data will be considered. By fit, we want a solution that represents the data as close as possible. However, one of the challenges in computer science is to determine whether we have the best solution possible: it is usually computationally infeasible to search the complete state space and ensure that we return the best solution possible. Thus, our search will by necessity be a heuristic process. In turn, this introduces bias as different search algorithms implement search in different ways. For example, when we build a decision tree, we make decisions on when to branch on a given attribute: doing this at an early stage may turn out to be ill-conceived based upon the subtree below that node.⁷ But probably the most common form of search bias that we encounter depends on whether we begin with a general description of our goal and refine it or start with a specific description and try to generalize it. The former is referred to as a *general-to-specific* bias, the latter as a *specific-to-general* bias. We shall see examples of both forms of model in this book and consider this type of bias within each context.

In our above discussion, we mentioned that our goal is to get the best fit we can to our data. In order to accomplish this, we can use many techniques, including domain knowledge and other heuristics. We might, for example, begin by focusing on the simplest data/solutions and build up to including more and more complex solutions in our model. We can stop when we reach a sufficiently complex solution and then review our data to determine how we move forward, introducing an iterative approach to the problem. However, we can go overboard and quickly find that our solution is, in fact, the best solution – but the best solution only for our current dataset. Generalization of our model may not be possible. Although this issue is an example of bias, it is sufficiently general and common that we consider it separately in Sect. 2.16 below. We further examine the *bias-variance decomposition* concept in Sect. 2.19 below.

One of the strengths of data mining is that we can test a large number of hypotheses about a particular dataset in a very short time. Exhaustively searching the dataset using combinations of variables in the hope that they might show a correlation is a factor of the processing power and size of the dataset; our ability to perform such an exhaustive search increases almost daily. But this is also a weakness. Statistical significance testing, based on the probability that an observation arose by chance, would indicate that it is reasonable to assume that 10% of randomly chosen hypotheses will be significant at the 10% level, 5% will turn out to

⁷This particular problem has been well studied in the automated reasoning realm. See, for example, Gallier (1986 #86).

be significant at the 5% level, 1% at the 1% significance level, 0.1% at the 0.1% significance level, and so on by chance alone. We can, therefore, safely assume that if we test enough hypotheses, some of them will appear to be highly significant – even if the dataset contains no real correlations!

This last concept might be best referred to as *miner bias* since we want to find patterns in the data and will often do so. However, there are many cases of published correlation results which subsequently turn out not to be correlations at all. Not every pattern has real meaning. One case, from the financial sector, has become infamous: David Leinweber “sifted through a United Nations CD-ROM and discovered that historically, the single best predictor of the Standard & Poor’s 500-stock index was butter production in Bangladesh” (Peter Coy⁸). We must be careful to validate that our mining results really **do** have merit.

2.9 Generalization

Generalization, the ability to leverage our models outside of the initial domain for which they were defined, is an important topic. We have already raised this concept in our discussions so far, for example, when considering generalization as search, and will return to it at various points throughout this book.

Leveraging a model to other datasets and, potentially, even to other domains provides a significant value for any organization. But while we have an intuitive understanding of what we mean by generalization that will serve us well, we also have to consider *how general* is general enough?

We shall see that one outcome of considering generalization when mining data is that complex models are often penalized; the more complex a model, the more likely it is that the complex model could not be effectively leveraged to other datasets and domains. Thus, we consider the predictive power and performance of any model we build. This is a primary factor in penalizing complex models.

2.10 Data Characterization and Discrimination

Data characterization is summarizing data of the class under study. Specifically, we are interested in the general characteristics or features of the data. These summaries may be based on simple statistical measures, or cube-based roll-up along a specified dimension.

Data discrimination is comparison of the target class with one of a set of comparative classes.

⁸Peter Coy’s BusinessWeek article “He Who Mines Data May Strike Fool’s Gold,” BusinessWeek June 16, 1997.

2.11 Association Analysis

One area of significant interest is to understand the relationships (affinities) between entities and variables in our dataset. Association analysis provides us with a set of techniques to model and analyze what these relationships may be. By what means these relationships occur is of secondary consequence to us: it is primarily *if* they are related on which we focus.

One area in which this set of techniques has been used to great advantage is in the area of disease-marker association.⁹ These efforts have shown, for example, a close association between the E4 allele in APOE and Alzheimer's disease: this allele is three times more frequent in Alzheimer's patients than in healthy control patients.

As we shall see when we consider association analysis in more detail later, we are interested in identifying direct, *causal* effects between entities/variables, indirect *correlated* effects, where an intermediary entity/variable may provide the link, but also to identify where associations may *appear* to exist, but which in fact may be false positives in our dataset. This is as valuable as identifying real associations since we can safely eliminate these associations from our dataset and ensure that our conclusions are as valid as possible.

2.12 Classification and Prediction

The processes of *classification* and *prediction* form a significant base for many data mining operations and constitute an important, fundamental technique. Note the singular form: classification is normally associated with categorical data, whereas prediction is the equivalent for dealing with continuous valued data.

The central idea of classification is to define a *model* that adequately describes the data provided in a *training set*. The data will partition into several classes,¹⁰ and the training set will include the concept labels along with the input data. We use the *trained model* to predict the concept associated with inputs that we have not previously seen and for which concept labels are unknown.

As stated above, prediction refers to continuous data as opposed to categorical data, and, instead of labels, we are aiming to predict missing or unavailable *numerical* data. *Regression analysis* (see later) is often used for numeric data prediction, but as we shall see, this is only one of many techniques we can use. In fact, a wide range of different techniques are used by researchers to predict data, including support vector machines, Gaussian processes, Bayes' nets, and neural networks.

⁹ http://bioinformatics.well.ox.ac.uk/lectures/Association_260404.ppt

¹⁰ Alternative terms include categories or concepts as used in Witten and Frank (2005).

As indicated above, classification is often used with categorical data and prediction is used for continuous valued functions and data, we use the term classification to cover both cases within this book when describing general principles and underlying theory.

2.12.1 *Relevance Analysis*

As a precursor to performing classification or prediction, it is often useful to try and identify any attributes that do not contribute to the process being performed through *relevance analysis*. In such cases, the attributes can be excluded from the process. This is a very powerful technique we can use at the beginning of our mining effort; if we can simplify our dataset by excluding irrelevant data attributes, our algorithms will be more efficient since they do not have to process as much data, they will not be adversely affected by considering irrelevant data to be relevant, which could then bias our results, and enhance our ability to intuitively understand the results generated.

However, the problem of excluding attributes which are, in fact, relevant can lead to erroneous inferences being made. Care must be taken when performing relevance analysis to ensure that we **really** understand an attributes' relevance.

2.13 Cluster Analysis

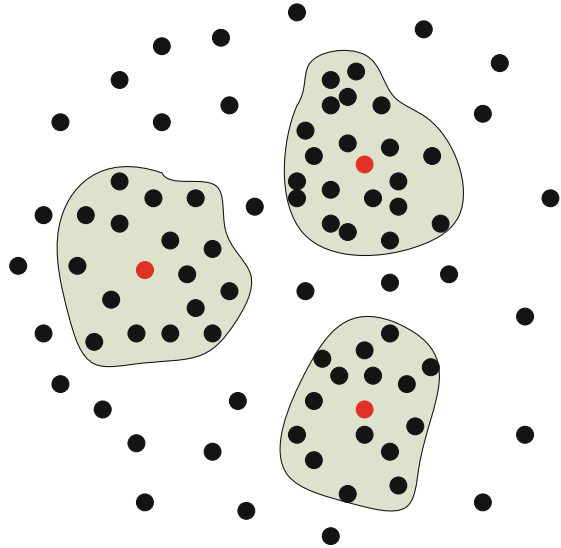
On many occasions, target labels for the input data are not known. In such cases, we may not be able to generate a training set for our model. However, this may not be of particular issue: we can use the method of maximizing similarities between inputs in the same class and minimizing the similarities between inputs in different classes. As stated in Han and Kamber (2006),

...the principle of maximizing the intraclass similarity and minimizing the interclass similarity.

What we need to do is generate clusters of objects where the members of that cluster have high similarity¹¹ in comparison to one another, but which are dissimilar to objects in other clusters. As indicated by the vagueness of *similarity*, this can cover a wide range. In fact, multiple similarity measures may be applied to a single dataset depending upon the perspective of the user and the mining intent.

¹¹ The concept of similarity will be a common theme through much of this book. If there is a fundamental concept for data mining, it is to identify things which are similar and things which are dissimilar. The concept of similarity, however, as we shall see later, can be defined in a wide range of terms.

Fig. 2.1 Two-dimensional cluster plot



We could have a similarity measure that identifies different groups of patients who have exhibited a particular reaction to a therapy. The “patient” data in Fig. 2.1 depicts how these different clusters may arise. The red spot indicates the cluster’s center measure.

2.14 Outlier Analysis

We introduce another premining technique that is often a necessary part of the mining effort. In the real world, we will often encounter datasets that contain data that does not conform to the model we define, or to the general behavior of the data. For example, our blood pressure measurements may include a value or two that cannot be valid if the patient is, in fact, alive. In such cases, we will discard such *outliers* as exceptions, or consider them to be noise in our dataset. However, in certain cases, we may be more interested in these exceptions than in the data which conforms to our model.¹² We often identify such outliers by using statistical methods that assume the dataset as a whole conforms to a probability distribution or model, or by using distance measures to highlight data points which are a substantial distance away from any of the clusters identified in our model. Another method is to look at the most interesting and important characteristics of the data in our dataset and identify any which deviates from these characteristics, by some substantial amount, as outliers.

¹² A good example of this is in the area of fraud detection, which is completely outside the scope of this book.

Outliers can therefore be a nuisance, or a value, depending on our perspective, but in either case, they constitute a factor that we need to consider: even some of our basic measures, such as the mean of a data set, can be affected by outliers.

2.15 Normalization

Normalization transforms continuous or discrete values or ranges to numbers, converting individual attribute values in such a way that all attribute values lie in the same range. Normally, values are converted to be in the range 0.0 to 1.0 or the range -1 to $+1$ to ease in further processing, ensuring that attributes do not receive artificial weighting caused by differences in the ranges that they span.

This technique is of particular value when using any statistical technique since a fundamental assumption is that a probability ranges between 0 and 1. But it also helps in many other mathematical algorithms since they will typically be well defined in this range. Further, we can be more comfortable with our analysis since all of our scales will be the same. This avoids any problems that might occur as we try and compare or associate variables that have different scales of values.

2.16 Dimensionality

All data mining algorithms work on a set of *data points*. These data points may have associated output labels, in which case we can take advantage of supervised learning algorithms such as hidden Markov models, neural networks, and the like, or they may not, in which case we use unsupervised learning algorithms such as k-means, SOM, etc. In either case, we typically represent each data point as a vector in a space (continuous or discrete as appropriate). We define the *dimensionality of a dataset* as the length of the vectors in the dataset. One *feature* per dimension of the dataset is another common term, where we define feature as being a data attribute of interest.

For example, consider the type of information we may feel it necessary to capture for a patient during each encounter (visit) for a study. This data could include:

- Age
- Height
- Weight
- Blood pressure
- Heart rate
- 30-panel clinical test results

Even here we have a total of 35 attributes; each vector within the dataset would have 35 attributes, giving us a 35-dimension dataset.

But a question we need to ask is whether or not every attribute is actually needed for the algorithm we are using. The algorithm will try and identify some of the parameters to model the distribution of the data points. Thus, the dimensionality of the parameter's space will necessarily be proportional to the dimensionality of the dataset. But this in turn means that the number of independent data points must be at least equal to, or greater than, the number of parameters needed for the model.

It is therefore important to carefully consider how the data points will be converted to vectors. This will have significant impact on the algorithm's success. Let's consider this issue in more detail. In many cases, converting the input data points, vectors in their own right, to vectors in some other form may increase the probability of an algorithm successfully processing the data. Thus, as well as leaving the data points as is, we can look to converting them to improve the chances of success. In kernel learning terms, this is where the input space is converted to a *feature space*, which we discuss later.

One method we can employ is to ask the question: do we have any attributes (dimensions) which are incidental to our goal? If we can identify such attributes, we can afford to reduce the dimensionality by excluding those attributes.

However, we must be careful with whatever approach we take. If we have too many dimensions, our model will require too many parameters, and therefore, more data points than we have in our data set. There is a danger that we may *overtrain* the algorithm and, when we try and generalize outside of our training set, that the model performs badly. Conversely, if we are able to reduce the dimensionality, we may end up with too few dimensions, in which case we may lose information that may be essential to identify the pattern, class, or other feature we are interested in. Bellman (1961) first coined the term “curse of dimensionality” which refers to the exponential growth of hypervolume as a function of dimensionality.

Another question we need to carefully consider is whether any of the attributes essentially duplicate other information. For example, if we have the date of birth of a patient, we need not capture the age (or vice versa).

As we consider mappings from our input space to an output space, our model needs to be able to cover or represent every part of the input space in order to know how that part of the space should be mapped (to its representative output space). The amount of resources necessary to accomplish this is proportional to the hypervolume of the input space.

Thus, the curse of dimensionality comes from the fact that models with lots of irrelevant inputs behave relatively badly. For models where the dimension of the input space is high, the model implementation uses an inordinate amount of its resources to represent irrelevant portions of the space. Many unsupervised learning algorithms are particularly susceptible to this problem.¹³ One way to alleviate this is to preprocess the input by normalization, scaling the input attributes according to some level of importance (so that some of the irrelevancies are eliminated), or other techniques which we

¹³ As radial basis functions (RBFs) that we consider later under Gaussian methods in learning.

will explore later. However, the higher the dimensionality of the input space, the more data which may be needed to find out what is important and what is not.

A priori information can be very helpful in managing the dimensionality issue. Careful feature selection and scaling of the input attributes fundamentally affect the severity of the problem as well as selection of the model.

Although we can use *guesswork* when considering what dimensions are necessary, there are several formal methods which can be valuable. *Principal component analysis* (PCA) rotates the dataset in such a way as to maximize the dispersion across major axes. *Singular value decomposition* (SVD) reduces a dataset containing a large number of values to a dataset containing significantly fewer values, while still maintaining a large fraction of the variability present in the original data. Both techniques are closely related to each other and offer significant value in managing the dimensionality issue, as we shall see later.

2.17 Model Overfitting

In any dataset sufficiently complex to be valuable for analysis, we will typically begin, and maybe end, with some form of simplification, or *modeling*, as we've already discussed. In building our model, we will encounter many points at which we have to make decisions that could dramatically affect the results of our analysis. Consider a set of data points, as depicted in Fig. 2.2.

Although both the curve and the straight line might be valid models of the data, most of us would agree that the straight line would be the better model due to its

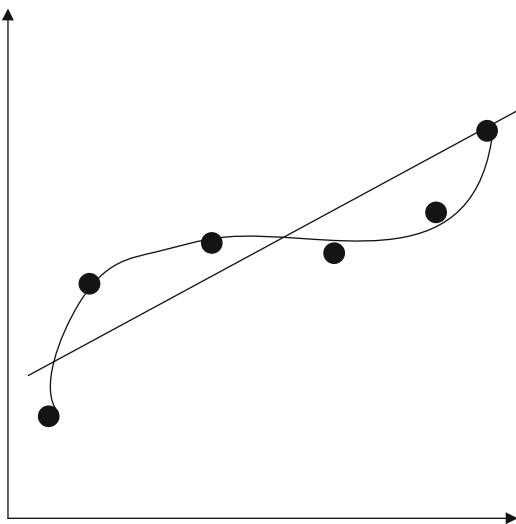


Fig. 2.2 Overfitting

simplicity and applicability.¹⁴ Further, as new data comes in, rerunning our model to recreate the straight line will be much simpler than to recreate the polynomial (hopefully!?) associated with the more complex curve. This being said, the curve would give a more accurate estimate/value for any point in our dataset.

This is a good example of the concern we expressed above: we can easily fall into the trap of developing a model where the architecture and parameters have been so tuned to the data we have used for training that its performance against a more generalized base of data is severely and negatively impacted. This situation can occur because of a number of factors:

- The features (characteristics) used may not be specific enough – or may be too specific.
- The model architecture may not be too sensitive, and not specific enough, or vice versa.

Overfitting occurs, therefore, when anomalies in the data presented to the model are incorporated into the model. That is, any *training data* that is used will naturally contain characteristics which are not necessarily in the general universe of data we will encounter once our model is used beyond its training stage, and we must ensure that when we validate our model, we do so using a dataset that is *independent* of the training data.

An overfitted model will obviously not make accurate predictions on unseen data. How can we ensure that the model is not overfitted? One approach we can take to determine whether a model M_1 is overfitting a training set is to identify an alternative model M_2 where this second model exhibits higher training prediction and lower test prediction errors with respect to M_1 .

Another approach is to consider a separate *validation dataset* that is independently selected and used once the model's training period has been completed. This can be used to determine initial overfitting, but is also a valuable technique which can be applied to the model periodically to see if the model needs to be tuned and/or trained to improve its generalization.

2.18 Concept Hierarchies

A *concept hierarchy* is any hierarchically organized collection of domain concepts, where the organizing relationship is the “part-of” relationship. One such example that we each deal with on a daily basis relates to the geographical location concept hierarchy where we have cities as a part of counties, which are parts of states or provinces, which are parts of countries. The Linnean binomial scientific classification

¹⁴ Would most of us agree with this contention? Certainly, a straight line provides a simplicity of representation, and it also provides a wealth of techniques based on relatively simple mathematical concepts that we began to learn in high school, but a pertinent question to ask at this point would be whether the straight line is the *most appropriate* model. Note that we avoided the use of the word “best”!

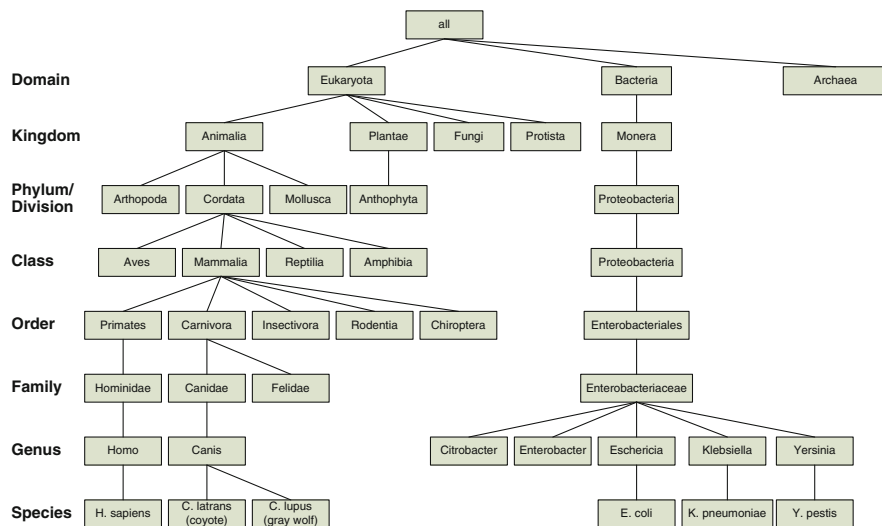


Fig. 2.3 Concept hierarchy fragment for Linnean binomial classification system

system also provides an example of a concept hierarchy, a fragment of which is illustrated in Fig. 2.3.

As we shall see in Chap. 3, our database schema will often implicitly contain many concept hierarchies through parent-child relationships.

Implementations of concept hierarchies will usually have a *total ordering*, or a *partial ordering* (also referred to as a lattice). A total ordering is probably most clearly seen by using the geographic location example referred to above. We may be interested in capturing information about the specific address (street and city), the state or province, postal code or zip code, and the country. We can easily see that there is an ordering where “street is within city, city is within state/province, state/province is within country.” Using the standard relational operators, this is often written as “street < city < state/province < country.” Here, we can see that each element on the left of the greater than symbol is completely contained within the element on the right of the greater than symbol. Our geographic location ordering is often physically implemented as a data *dimension*, which we discuss in more detail in Chap. 3.

Another dimension, ubiquitous to data warehouse environments, is the time dimension. This is a good example of a partial ordering. Consider that we typically think of time from some fragment of a second – say the millisecond – up to some larger interval – say the year. Our ordering would then be “millisecond < second < minute < hour < day < week < month < quarter < year.”¹⁵ However, if we look carefully, we can see that this is not a totally accurate ordering, since a week may well start in 1 month and end in the next. For this reason, we typically begin to break this

¹⁵ This ordering covers almost all of the time intervals that *any* data environment would be looking for! Decades, centuries, millennia, and beyond are outside the scope of this book?!

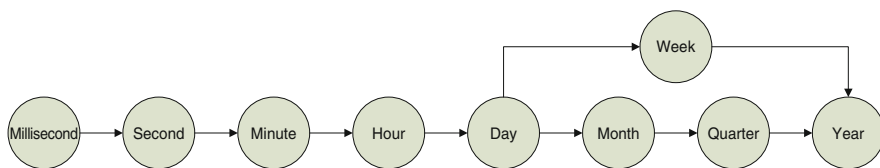


Fig. 2.4 Time concept hierarchy

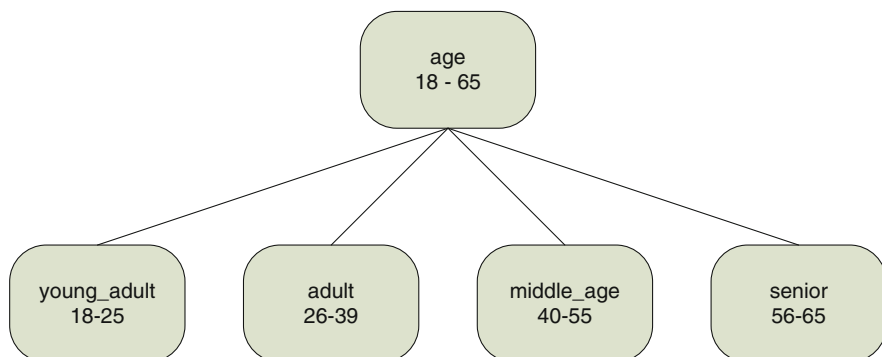


Fig. 2.5 Age concept hierarchy

ordering into several *partial* orders. The “millisecond < second < minute < hour < day” ordering does meet our definition of a total ordering. It is when we look at the “week < month < quarter” ordering that things change. A week may span months, but it may also span quarters. So we really have three options to link the day to the year: “day < year,” “day < week < year,” and “day < month < quarter < year.” Diagrammatically, we can represent this set of hierarchies as shown in Fig. 2.4. Hierarchies such as this will often be embedded in the data environment.

We can create concept hierarchies for any given dimension or attribute (variable) by grouping values or discretizing the values. An example we use elsewhere in this book is that of age. For a large-scale, international, phase III clinical trial, we may have patients of every age between 18 and 65. We may not really be interested in discretely separating all 48 ages; we may instead be interested in patients who are *young adult* (18–25), *adult* (26–39), *middle age* (40–55), *senior* (56–65) (Fig. 2.5).

This is a particularly simple example of a concept hierarchy. We may want to define multiple hierarchies for a single attribute. For example, we may be interested in segmenting the age by decade (18–19, 20–29, 30–39, 40–49, 50–59, 60–65).

The key point is that concept hierarchies allow us to analyze data at multiple levels of abstraction. While we typically want to consider data at the lowest level of granularity, being able to abstract some elements of the data can be very valuable, if for no other reason than our being able to get a sense of where the data and model are leading us.

That being said, we also have to be careful and ensure that our abstractions make sense. While there are obviously similarities between all members of the *Enterobacteriaceae* family, for example, we must ask ourselves whether our model and results will make sense, or whether we should really be looking at the *Escherichia* genus or *E. coli* species.

2.19 Bias-Variance Decomposition

Any model we develop will have an inherently built-in error component inherently due to noise, missing data, assumptions, and other considerations we have discussed elsewhere. We therefore want to try and calculate the expected error of our model for a given target result (concept, pattern, correlation, etc.) and a given training dataset size. This expected error can be broken out into two parts: a bias component and a variance component.

One of the important factors of this decomposition is that it is valid for finite training sets and not just in the asymptotic, or limiting case. Thus, we can measure each of the terms experimentally.

The bias component measures how closely our model's results match the target. Specifically, we measure the average guess over all possible training sets of a given size and correlate these with the target. The variance component measures the variation of the model's output for different training sets of the specified size.

Unfortunately, trying to improve one of these components on its own often leads to frustration. One approach often used is to increase the state space for the model in order to reduce the bias component. This will often reduce the bias term but increase the variance term. For example, Kohavi (2000) shows a subset of data that has a reasonable fit using a quadratic and a perfect fit using a 9th order equation. Since every training set will inevitably generate a different 9th degree polynomial, the result will be a high variance component.

Models that fit a linear model will typically be biased for quadratic and higher-order models, but for several algorithms, regularization techniques can balance this trade-off.¹⁶

2.20 Advanced Techniques

We are going to take a short sidebar and introduce two concepts that some readers may be familiar with, but which we'll return to later in the text. For this reason, it is safe to skip this section.

¹⁶ We will touch upon this subject a little further in later chapters, but a full consideration is outside the scope of this book. The interested reader may find Kohavi (2000) or Duda (2001) valuable starting points for a further understanding.

However, *principal component analysis* (PCA) and *Monte Carlo* strategies provide good illustration into how we can incorporate tools, techniques, and concepts from other disciplines to help us in our analysis efforts.

These two techniques are only illustrations. There are myriad other techniques, some of which we explore in this text, that provide novel solutions using novel approaches. This is one of the exciting things about data mining, and the reader is strongly encouraged to keep abreast of research endeavors of this dynamic field. You never know when a technique used in a novel way might be just what you are looking for.

2.20.1 *Principal Component Analysis*

Principal component analysis (PCA) is a valuable statistical technique which is used in a wide variety of fields and is a common technique for finding patterns in high-dimensional data and to express the data so as to highlight their similarities and differences.

High-dimensional data can be a challenging analytical problem since we do not have the luxury of a visual representation to help us. But using PCA to find patterns subsequently helps with this problem since once patterns have been identified, we can reduce the number of dimensions (compress the data) without losing information.¹⁷

Given a set of data, we can use the following process to perform PCA:

1. Subtract the mean from each data dimension.
2. Calculate the covariance matrix for each pair of dimensions.
3. Calculate the eigenvectors and eigenvalues for the covariance matrix.
4. Order the eigenvectors by eigenvalue (largest to smallest) to give the principal components.
5. (Elect to) ignore components with small eigenvalues (loss of information **should** be minimal).
6. Derive the new data set.

2.20.2 *Monte Carlo Strategies*

Any reader familiar with Monte Carlo strategies will probably be wondering why we would describe an advanced technique at this point in the text. We will be considering Monte Carlo strategies in significant depth later in this book, but introduce and illustrate the concept here for completeness and because it provides a good example of how some of these more advanced stochastic techniques are providing tremendous value in the analysis of life science data through their ability to provide a modeling framework, allowing simulation of physical systems.

¹⁷ This is particularly useful in image management.

The nondeterministic nature is typically apparent through the use of random numbers¹⁸ as input to the methods. In addition to this, these methods are characterized by repetition of the algorithm – often a large number of repetitions – and a (very) large number of calculations. These characteristics make them well suited to the computer environment. One factor that has made such algorithms very attractive in many areas of scientific research has to do with their relative efficiency, when compared to other methods, as the number of dimensions increases.

For readers interested in a little more detail, please see the box below. Those readers not interested in further explanation of the theory at this time may skip the box without a loss of understanding.

In data mining, and in scientific disciplines in general, our region of interest will often be a high-dimensional space, along with some target function within that space. Mathematically, we often model this as the computation of an integral, I :

$$I = \int_{S_d} f(x) dx,$$

where S_d is our space with dimension d , and f is the function we are interested in.

However, such integrals may not always be easy to solve, and so we undertake to use some computational techniques to compute an estimate for I , denoted by \hat{I} . A prerequisite for this is being able to select independent and identically distributed random samples from our dataset. Using the *law of large numbers*, which states that the average of a large number of such random variables with common mean (and finite variances) will tend to their common mean, we can obtain an estimate as close to the real value as we like by increasing the number of iterations we perform.

Where problems exist that involve large degrees of freedom, or where there is significant uncertainty in the inputs, this class of method has shown itself to be particularly valuable. For example, the *cellular Potts model* is a model that simulates the collective behavior of cellular structures (Graner and Glazier 1992 #142).

Monte Carlo methods have been successfully applied to many areas both within the scientific discipline and without. Areas include molecular simulation, inference in population genetics, finding motif patterns in DNA, risk in business – especially in the insurance industry – as well as being very important in disciplines such as computational physics and physical chemistry. Other areas include graphics, modeling light transport in multilayered tissues (MCML), finance, reliability engineering, protein structure prediction, modeling transport of current carriers in

¹⁸ Really, pseudorandom numbers.

semiconductor research, contaminant behavior in environmental science, molecular modeling, counter-pollution models, simulation of atomic clusters, computer science, movement of impurity atoms and ions in plasmas, and particle physics. Their use in computational mathematics has been varied, such as in primality testing (Rabin's algorithm (Rabin 1980 #143)). In addition, Monte Carlo methods have been used to help with the missing data problem.

We include a brief description of a very widely known experiment – “Buffon’s needle” – that illustrates the basic ideas of Monte Carlo strategies.¹⁹ In this experiment, a needle of some predetermined length l is dropped onto a flat surface that contained a square grid with spacing D , where $D > l$. Under ideal conditions, the probability of the needle intersecting one of the lines is $\frac{2l}{\pi D}$. If we let the proportion of times that the needle has intersected a line in n throws to be denoted by $p(n)$, we can compute an estimated value for π , denoted by $\hat{\pi}$ to be

$$\hat{\pi} = \lim_{n \rightarrow \infty} \frac{2l}{p(n)D}.$$

As the above equation illustrates, and the previous discussion highlights, generating random samples from our probability distribution is a fundamental step in using Monte Carlo methods.

As an illustration, let us generate an estimate for the value of π . This is a very widely used, simple example of a Monte Carlo method and can be found in many places in the Internet. If we wish to calculate a value for π , we can use a circumscribed unit circle within a square and model the algorithm below using the whole circle. However, we can simplify our model a little by considering only a single quadrant, with radius r , inside a square of radius r .

If we were to throw darts at the figure, we can see that

$$\frac{\text{\#darts in quadrant}}{\text{\#darts in square}} = \frac{\text{area of quadrant}}{\text{area of square}}$$

Or, from geometry,

$$\frac{\text{\#darts in quadrant}}{\text{\#darts in square}} = \frac{\frac{1}{4}\pi r^2}{r^2} = \frac{\pi}{4}$$

which gives us

$$\pi = 4 \frac{\text{\#darts in quadrant}}{\text{\#darts in square}}$$

We can implement this algorithm very simply and effectively, as the following R function shows.

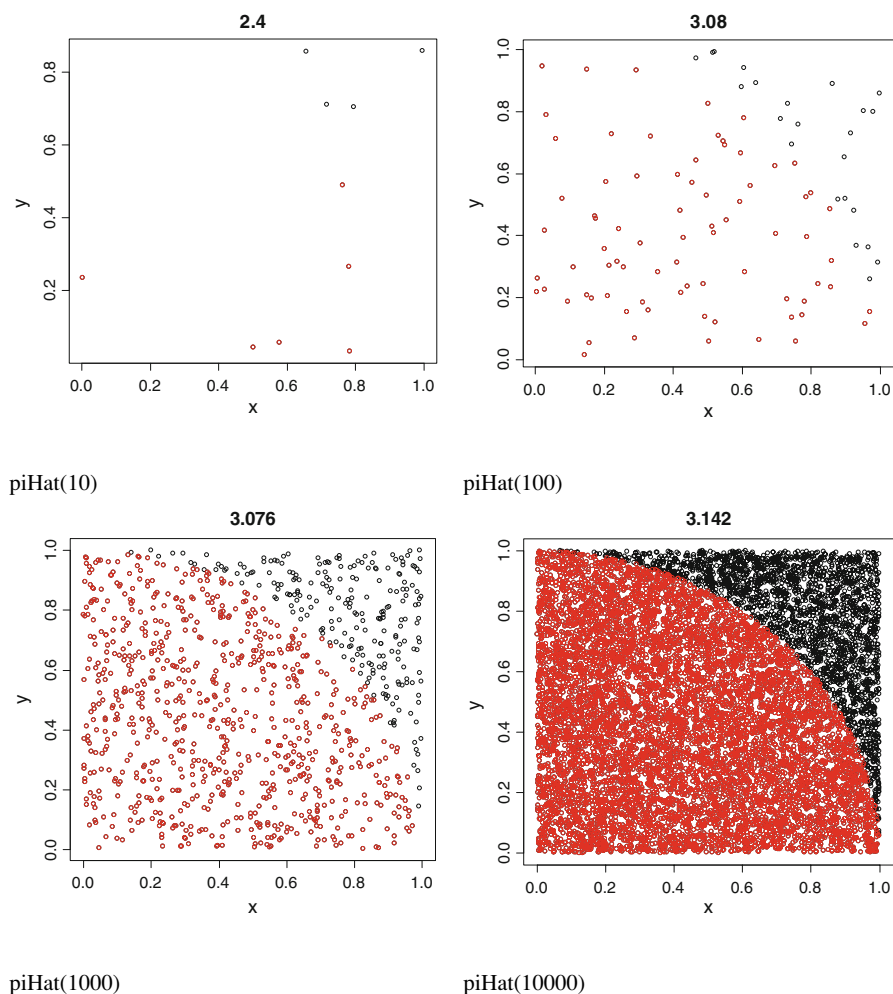
¹⁹ Louis Leclerc Comte de Buffon, 1777.

```

piHat <- function(N) {
# N <- 1000
x <- runif(N)
y <- runif(N)
idx <- x^2 + y^2 <= 1.0000000000000000
piHat <- length(which(idx))/N*4 # estimate of pi
plot(x,y,main=piHat)
points(x[idx], y[idx], col='red')
sprintf("Estimate of pi = %.9f after %d iterations", piHat, N)
}

```

The parameter N is the number of observations which should be used for generating random values from the uniform distribution (`runif`). Running this function for four values provides the following output:



The above examples also illustrate one of the key attributes of Monte Carlo methods: a large number of repetitions are often necessary to generate good estimates. Commenting out the plot and points functions in piHat provides the following estimates for a number of different values of N :

```
> piHat(10)
[1] "Estimate of pi = 3.600000000 after 10 iterations"
> piHat(100)
[1] "Estimate of pi = 3.320000000 after 100 iterations"
> piHat(1000)
[1] "Estimate of pi = 3.024000000 after 1000 iterations"
> piHat(10000)
[1] "Estimate of pi = 3.162400000 after 10000 iterations"
> piHat(100000)
[1] "Estimate of pi = 3.140360000 after 100000 iterations"
> piHat(1000000)
[1] "Estimate of pi = 3.141168000 after 1000000 iterations"
> piHat(10000000)
[1] "Estimate of pi = 3.141554800 after 10000000 iterations"
```

2.21 Some Introductory Methods

Now...back to our program already in progress...

To conclude this chapter, we highlight some simple methods that provide surprisingly good results even though the methods are very simple. In this section, just as was the case above, we'll introduce them, but go into more detail later in the text.

These techniques provide a good heuristic for any analysis effort: using as simple a technique as possible has many benefits. Often the simplest approaches provide not only the valuable outputs in their own right but also a mechanism for targeting subsequent mining activities. This is not to be exhaustive by any means, but simply an attempt to introduce techniques that are widely used and which are typically some of the first to be encountered.

2.21.1 *The 1R Method*

The 1R (for 1-rule) method outlined in (Witten and Frank 2005) considers a single attribute in each instance of our dataset and develops rules based upon this categorization.

The way in which this method works is to look at a single attribute and count the number of times that each value, or class of values, occurs. Assign the most frequent (class) value to this attribute (as the default) and then calculate the error as being the

number of times the attribute has a value other than the most frequent value. Repeat this for each attribute and then select the rules with the smallest error rates.

For each attribute in our dataset

For each value of the attribute

Count the number of times each value (class) occurs

Assign the most frequent value (class) to the attribute

Calculate the error rate as the count of all other values

Select the rule(s) with the smallest error rates

Consider the following table of (made-up) data:

Organism	Compound	MIC	Interpretation ²⁰	Safety	Efficacy
SP001001	C00100101	2.0	I	Medium	Yes
SP001001	C00100102	1.0	S	High	Yes
SP001001	C00100103	4.0	R	Low	No
SP001002	C00100101	1.0	S	High	Yes
SP001002	C00100104	0.5	I	Medium	No
SP001003	C00100101	2.0	I	Medium	Yes
SP001003	C00100102	2.0	I	Medium	Yes
SP001003	C00100103	0.125	S	High	Yes
Sp001003	C00100104	0.25	S	High	Yes

When there are equal numbers of values that can be selected as the assignment (i.e., where there are an equal number of errors), we randomly make a choice. For the above example dataset, we will classify on “efficacy,” and the rules we define, with the smallest error rates, are as follows:

Attribute	Rules	Errors	Total errors
Safety	High → yes	0/4	1/9
	Medium → yes	1/4	
	Low → no	0/1	

The “rules” column shows the default (majority) value. In the first case, a “safety” value of “high” maps to an efficacy value of “yes” more often than any other value. The same is said for the other two rules. The “errors” column then displays the number of times that the rule **does not** occur; for the “medium → yes” rule, this is once (row 5).

²⁰This table contains example microbiological data where the interpretation is determined according to the breakpoints issued by the FDA and/or CLSI for a particular organism-compound combination in a given issuance period (typically a year).

2.21.1.1 Missing Values

For missing values, we can often look to the class of values and add a value which corresponds to missing data, thus explicitly incorporating it into our dataset, while similarly partitioning the attribute’s value domain. This can be immensely valuable as we can immediately see what impact, if any missing values have.

Consider a dataset where the cholesterol level is not provided for several patients, and for the rest, we have nominal values of critical, very high, high, normal, and low. Adding the nominal value “missing” now gives six different values for the attribute.

Using the 1R method, we may find that “missing” forms a significant number of errors in our algorithm. What should we do? In the extreme, what do we do in the event that this is the most frequent value?

We can also consider our dataset *excluding* any instances with missing values. Oftentimes, performing the analysis by including and then excluding such instances will provide us with significant insight into the dataset itself since we may find that the missing data as little to no impact on our analytical results. However, the converse could also be true.

An alternative would be to look at the attribute’s domain of values and assign missing values to one of the existing values rather than create an additional one. Using our cholesterol value domain above as an example, we could assign the value “normal” to our missing data. Obviously, this is dangerous since assigning such a value must be done with a complete understanding of its impact. Would we arbitrarily inflate the number of people with “normal” values? The answer is a definite yes if we are counting. However, for identifying patterns, we may find this a valuable technique.²¹

2.21.1.2 Numeric Data

Although not explicitly identified above, the 1R method provides most of its value when dealing with discrete data values which can be categorized rather than dealing with (raw) numeric data. However, numeric data can be converted to nominal data to maximize the value of this method.

One way of doing this is to use the concept of discretization. Consider a dataset where we have captured the patient’s age. We may have a dataset as follows, where the yes/no values indicate whether therapy was considered successful at the test of cure encounter.

22	24	27	29	31	34	37	41	42	49	55
yes	no	yes	yes	yes	no	no	yes	no	no	no

²¹ We introduce this idea here but will return to it at different times in this book as its value waxes and wanes with the different techniques we consider.

We now place a breakpoint in the data. The most intuitive way is to place a breakpoint in wherever the class changes, as follows:

22	24	27	29	31	34	37	41	42	49	55
yes	no	yes	yes	yes	no	no	yes	no	no	no

which produces a total of six categories in this example. We can then place a breakpoint midway between each pair delimiting a category change. However, this would cause a problem if we had two patients, each aged 34, one of whose therapy was successful, whereas the other was unsuccessful. In such cases, we can move the breakpoint up to the next partition (between ages 34 and 37) and consider a single partition with a mixture of outcomes.

However, this is not the only approach we can take. For many of these issues, we must consider the dataset in question and the objective of our analysis; a different approach may be more appropriate.

2.21.2 Decision Trees

Decision trees are arguably one of the most widely used techniques in data mining. In many disciplines, we are familiar with the technique of “dividing and conquering” as a means of decomposing a problem into increasingly simple subcomponents until we reach a point at which each component can now be (easily) solved. A natural way to represent such decomposition is a tree structure.

Each *node* in our tree structure represents the testing of an attribute and is usually a test of the attribute with a constant value. For example, we may have a node in our tree that tests a patient’s cholesterol level against the upper recommended value of 200 mg/dL (Tietz 1995) and branch based upon whether the patient’s value is above this value or not.²² We can continue to build our tree by introducing further subnodes to continually refine our tree until we reach a level containing *leaf nodes*, each of which gives a classification to all instances which reach this node. In the simplest case, this classification is a single classification, but as we shall see later, this can also be a set of classifications or a probability distribution over all the possible classifications which reach this node. Consider our naïve example of comparing a patient’s cholesterol against the upper normal range value (Fig. 2.6).

We see here that we classify a patient as either “normal” or “at risk.” Any new patient data that we receive, previously unknown, would be processed through the successive nodes of the tree, being tested at each level, until it reaches the leaf node which classifies it according to the classification of the leaf node itself.

²² As we shall see later, more complex trees can test two attributes against each other, or even use functions to effect these tests.

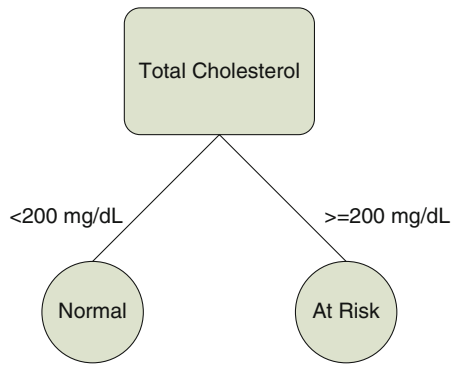


Fig. 2.6 Simple decision tree

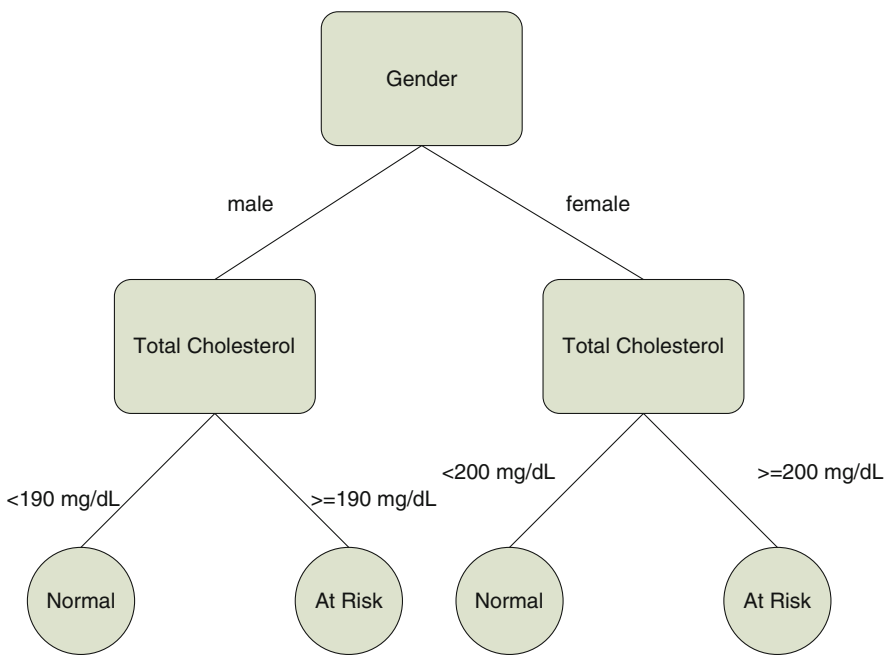


Fig. 2.7 Slightly less simple decision tree

The decision tree shown in Fig. 2.6 provides very little classification, and in real life, we would need to consider (many) more patient demographics to more closely model reality. The next step might be to consider that test result reference ranges are known to be different by gender, ethnicity, and age, for example. We might, therefore, expand our decision tree by introducing gender, to give us the tree shown in Fig. 2.7.

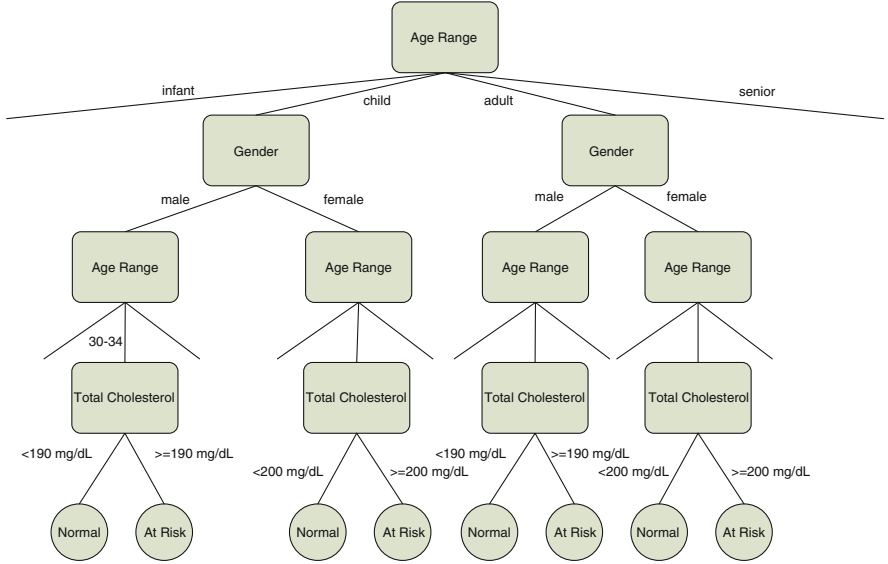


Fig. 2.8 Still simple decision tree

Here, we have decided that gender is the first decision point we need. Based upon the gender, we then test the total cholesterol value to classify the individual.²³ We see that our tree now contains a copy of the “total cholesterol” subtree. This often occurs, and we shall see how to better deal with this situation when we return to decision trees in more detail later.

In each node in our example trees above, we have had binary decision points. Trees can be generated with more than two branches. However, complexity also increases. If the attribute being tested is a numeric attribute, the test is usually a “greater than value”²⁴ type of test, giving a binary split as we have seen in our example. We may also use a three-way split, which has a number of additional valuable characteristics. We shall return to this later. Testing using the “greater than value” may not be valuable, especially at the initial levels of the tree, and we will often subset the attribute’s values. For example, we may not be interested, initially, in looking at a very granular breakdown of a person’s age, but may instead be interested to know if they are an infant (<2 years), child (2–16 years), adult(17–65 years), or senior (>65 years), where the parenthetical values represent our model of these age groups. In this case, a 34-year-old would be classified in the upper levels of the tree as being “adult.” However, when we look at that patient’s cholesterol level, we may use the partitioning in (Tietz 1995) (page 130) which has separate reference ranges for each 5-year increment in age. In such a case, the age would be tested once to partition into infant, child, adult, and senior and once again to put the patient into the 30–34 age range for our test. Thus, numeric attributes are often tested multiple times in a decision tree (Fig. 2.8).

²³ Note that we have changed the upper limit value for males. This is for illustration purposes only and should not be construed as of any medical significance.

²⁴ This is meant to stand for the group of binary tests which include <, >, <=, >=.

If the attribute being tested is a nominal attribute, the number of branches is typically the same as the possible values of the attribute. This may not be valuable, especially at the initial levels of the tree, and we will often subset the attribute's values.

Figure 2.8 illustrates another factor of decision trees which must be considered when designing the tree itself. We can see that subtrees are cloned in various places in the tree. These arise from two particular cases. The first relates to the *order* in which we decide to test attributes. In the above example, we elected to begin by testing the patient's age. Since we decided to partition the age into gross ranges first, we had to test it a second time further down the tree (as we described above). By choosing a different ordering, we may either change/reduce the cloned subtrees or even eliminate them together, although in real-world cases, this latter case is exceptional. The second case arises as a result of the *granularity* we are trying to achieve. Using the age test once again, is it really important to obtain the level of granularity greater than the infant/child/adult/senior partitioning?

An obvious problem with decision trees relates to missing values. If the dataset being studied is missing gender, for example, how do we handle the gender test in Fig. 2.7? If we cannot consider the missing value as a separate attribute value in its own right, we should process it in a special way rather than treating it as just another possible value for that attribute. Consider the age we decided to include in our decision tree of Fig. 2.8. What could we do if the age is missing for a number of patients? If we know something about the patients as a whole, we can use this information. For example, in a phase I clinical trial, we know that the patients will typically be healthy volunteers. We can make some assumptions about this type of demographic, especially if we are primarily looking for healthy adults – the 18–45 age range seems to be particularly popular at the time of writing, according to radio advertisements for clinical trials. Using this information, we could use a Gaussian probability function to “replace” the missing information. Alternatively, we could use another branch at the decision point. We shall return to both of these in more detail later.

Before we leave our discussion on decision trees, there is one limitation that needs to be highlighted that relates to using classification rules (see below) to generate a decision tree. Modeling the disjunction that is implicit in classification rules in a rule set is not a straightforward activity. We will discuss this further in the next section.

2.21.3 Classification Rules

In many respects, classification rules can be considered a direct alternative to decision trees. They are certainly, justifiably popular, and it is possible to move between these two techniques.

Classification rules can most easily be thought of as “if...then...” rules where we have an *antecedent* that specifies the preconditions for the rule to *fire*, and the

consequent (or conclusion) which specifies the conclusion of the rule – the results of it firing. Using the same model used to create the decision tree in Fig. 2.8, we might first begin with a classification rule as follows:

```
if total_cholesterol > 200 then risk_factor = high
```

In the above, we conclude that the variable `risk_factor` should be given the value of “high” if this rule fires. Instead, we can also define a probability distribution over the classes covered by this rule, as we shall see later.

The precondition section of a classification rule will often include several clauses that are logically ANDed together or may even be more complex logical expressions. For example, we might change our classification rule above to include gender as follows:

```
if gender = male and total_cholesterol > 200
    then risk_factor = high

if gender = female and total_cholesterol > 190
    then risk_factor = high
```

We have now developed two classification rules, one for each gender. For a rule to fire successfully, all of the clauses in the precondition must be true. The two rules illustrate another facet of classification rules: the *rule base* containing the set of rules (in our current example, the two rules) considers the rules to be ORed together. That is if any one of the rules applies, the class (or probability distribution) in the consequent is applied to the instance of our dataset we are currently considering. As the rule base grows in its number of rules, problems of conflict between the rules and circular dependencies become apparent.

We indicated earlier that we can easily move between decision trees and classification rules; this is true. In fact, we can usually create a classification rule for each leaf of the tree. The antecedent for each such rule will include a condition for every node on the path from the root of the tree to that leaf node. While the rules thus defined are unambiguous in our rule base, the rules tend to be more complex than necessary. However, we can easily prune the rules to remove redundant tests. One advantage of developing classification rules from decision trees is that execution of the rules can occur in any order.

Reversing the process – creating a decision tree from a set of classification rules – is not quite as straightforward as going from decision trees to classification rules. The primary reason for this has to do with our statement that the rules are considered to be ORed together (disjunction). This requires us to do make some decisions and perform some extra work to ensure that the resulting tree operates correctly. Let us consider a simple abstract example to illustrate this. Let us assume our rule set contains the following rules:

```
if a and b then X
if c and d then X
```

Logically, the above rules are equivalent to

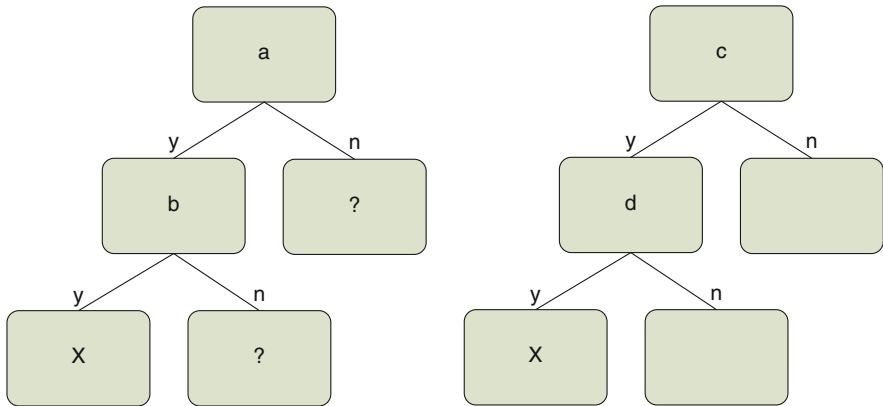


Fig. 2.9 Two subtrees in a disjunction relationship

if (a and b) or (c and d) then X

That is, if *a* and *b* are both true, *or* if *c* and *d* are both true, then we can validly make the conclusion that X is true.

But in order to accommodate this disjunction, we need to select a test for the root node of this tree fragment, and, as we shall see, we end up having to replicate part of the tree represented by these rules. This is known as the *replicated subtree problem*. Let us begin by visualizing the two rules we defined above in terms of individual subtrees (Fig. 2.9).

Since our intent is to combine these two subtrees into a single subtree, our first issue is to select the test for the root of the subtree. Intuitively, selecting to test “a” or “c” would seem to be a good starting point. Without loss of generality, we’ll choose to test “a” first. In doing so, our subtree looks exactly like the left-hand tree in Fig. 2.9. Where does the right-hand subtree fit in? It needs to be replicated in two places – shown in Fig. 2.9 by “?” – to provide a complete definition of our disjunction. The complete tree is shown in Fig. 2.10.

This problem can quickly cause it to become impractical to convert rules into decision trees. Imagine, for a moment that instead of two sets of binary conditions leading to a single conclusion, we have 3, or 4, or ... Now envision our rules having 3, 4, or even more conditions on the left-hand side!

Rules, themselves, have a long history in knowledge elicitation, artificial intelligence, and mathematical logic. We can envision each rule as holding some piece of valuable knowledge, and adding new rules will not necessarily perturbate the existing knowledge base. However, there is an issue that can arise from here that we need to consider: does adding new rules cause cycles or inconsistencies to occur? When our rule bases are small, we can visually inspect them to ensure that such cycles do not occur, but as our rule bases grow, such cycles may not be obvious.

A second problem that arises is due to the order in which rules are executed. Some environments will execute the rules in the order in which they are entered in the rule base. This leaves the problem of ordering to us. Other environments consider the ordering of the rules to be immaterial. Both cases highlight challenges that we need to be careful of when we develop our rule bases. In the former case, if

Association rules define or express the regularities underlying the dataset. For this reason, each rule predicts a different thing. Since we are *associating* two or more facets of our dataset together – the regularity – the number of association rules that can be elicited from even a small dataset can quickly become huge.

Let us consider our patient dataset.²⁶ From our first example dataset, we can identify the following association rules:

```
if (weight/(height2) > 39 then weight_category = extreme_obesity

if (weight/(height2) > 29 and (weight/(height2) < 40 then
    weight_category = obese

if (weight/(height2) > 24 and (weight/(height2) < 30 then
    weight_category = overweight

if (weight/(height2) > 18 and (weight/(height2) < 25 then
    weight_category = normal

if (weight/(height2) < 19 then weight_category = underweight
```

Let's pause to illustrate a challenge with association rules. As we can see from the above sets of rules, we have defined rules for a very small part of our dataset. In fact, the above five rules really identify a single conclusion, the body mass index. This situation of generating a large number of rules from a very small set of attributes (not the data itself) is common and can lead to an unwieldy set of rules if we are not very careful. We can use several techniques to help us to manage association rule, and we now consider some of the more common.

In order to keep the rules we elicit manageable, we define two characteristics:

- The *coverage* (a.k.a. *support*) of a rule is the number of instances from our dataset that the rule predicts correctly.
- The *accuracy* (a.k.a. *confidence*) of the rule is the number of instances from our dataset that the rule predicts correctly as a proportion of the number of instances to which it applies.

Suppose we have a rule in our set that can be applied to 50 instances in our dataset (its coverage or support) and would predict 10 of those instances correctly. We can calculate its accuracy to be 20%. The question we next need to ask is whether this is valuable or not. As we shall see later, we can set minimum values for these two characteristics and eliminate association rules which do not meet these criteria. For example, if our dataset contains 1,000,000 instances and a particular set of rules only applies to 50 instances and only correctly predicts 10 of those

²⁶ We will have cause to consider a dataset that needs to be more complex than some of our other datasets in order to provide some meaningful output; we will use our patient dataset for this purpose. See [Appendix C](#) for a full description of the dataset and for example data.

instances, we may eliminate this rule from our association set on the grounds that its applicability is too limited for our current purposes.

Association rules, as shown above, do provide a lot of value, but they need to be identified, and even crafted, carefully. For example, our set of association rules for body mass index would probably be eliminated on the grounds that they are really calculations that we might elect to implement as an additional attribute in our dataset instead. This would then allow us to use them across a wider range of applications. We could then use that attribute, together with other information to form a different set of rules, such as those shown below.

```
if weight_category = extreme_obesity then risk_factor = 8.0

if weight_category = obese then risk_factor = 4.0

if weight_category = overweight then risk_factor = 2.0

if weight_category = normal then risk_factor = 1.0

if weight_category = underweight then risk_factor = 2.0
```

Here, we have assigned a factor that can be used elsewhere in our rule set.

So far, we haven't discussed conjunctions ("and") and disjunctions ("or") in our rule set, but these natural extensions, along with negation ("not"), can expand the value of association rules. For example, National Football League players are more often than not in a higher than normal weight category, but it is questionable whether they would have a higher risk factor because of their BMI value. We might change the above rules as follows:

```
if weight_category = extreme_obesity and activity_factor < 8
    then risk_factor = 8.0

if weight_category = obese and activity_factor < 7
    then risk_factor = 4.0

if weight_category = overweight and activity_factor < 6
    then risk_factor = 2.0

if weight_category = normal and activity_factor > 2
    then risk_factor = 1.0

if weight_category = underweight and activity_factor > 2
    then risk_factor = 2.0
```

This is one of the reasons why our rule sets can become much larger, very quickly. We now need to ask ourselves whether or not the conditions not included in the above rule set are important or not. For example, just using the first rule:

```
if weight_category = extreme_obesity and activity_factor < 8
    then risk_factor = 8.0
```

The condition not covered is

```
if weight_category = extreme_obesity and activity_factor >= 8
    then risk_factor = ???
```

(Note that the other condition, where `weight_category != extreme_obesity`, is covered by other rules.) Is it important for us to include this additional rule? Unfortunately, the answer is “that depends.”

We can extend classification rules or association rules in some intuitively natural ways to increase their flexibility. One such way is to introduce the concept of *exceptions* to the rules. We may, for example, discover that there is a relationship between a patient’s total cholesterol and their LDL cholesterol such that if their LDL cholesterol is below a certain threshold, it reduces their risk.

```
if total_cholesterol > 200 then risk_factor = high
    except if ldl_cholesterol < ?? then risk_factor = moderate
```

Without this exception category, we would either misclassify the instance or would require additional rules, with the potential for causing other problems.

2.21.5 Relational Rules

The rules we have considered so far are called *propositional rules*,²⁷ but these are oftentimes insufficient for our needs. Such situations typically require us to be able to express the *relationships* between the instances in our dataset. For example, we may be interested in the relationship between weight and height. For these types of situations, we need more expressive rules.

Consider relating weight and height through the body mass index that we introduced in our discussion on association rules above. The formal definition of the BMI is

$$\begin{aligned} bmi &= \frac{weight(kg)}{height^2(m^2)} \times 100 \\ &= \frac{weight(pounds)}{height^2(in^2)} \times 703.1 \end{aligned}$$

²⁷ The language in which we express such rules has the same expressive power as the propositional logic.

BMI tables exist all over the Internet.²⁸ From the above equation, or the tables, we can easily see a relationship between height and weight. We may be interested in expressing this type of relationship within our association rules. For example, we can say that

```
if height < 65 and weight > 150 then risk_factor = high
```

to represent a baseline, since it appears that a 5' 5" person is more likely to weigh more than 150 lb in today's society than not. Similarly, we might elect to include a rule as follows:

```
if height > 74 then risk_factor = undetermined
```

since there is some debate as to how effective the BMI measure is at the tails of the population and for athletes, etc., as we mentioned earlier. For the same reasons we discussed for association rules, we might therefore want to introduce rules that cover such cases. We can use activity factor attribute in our patient dataset for this purpose, in which case the relational rules will look very similar to our association rules.

2.21.6 Clustering

The intuitive concept of clustering is to group similar instances together. This allows us to simplify our dataset – especially since we will typically be dealing with vast amounts of data – for further processing. However, the challenge here relates to the fact that as soon as we start grouping similar instances together, we lose the fine granular details that make each instance distinct and unique. Further, we need to be careful about our definition of similarity since we can easily end up with too few clusters – meaning that we have too gross a definition of similarity – and lose too much detail. The conclusions we then draw will essentially be naïve, or too simplistic, on the one hand, or of little value on the other.

As we shall see later, clustering is a widely used technique and has a rich history in the machine-learning community where *unsupervised learning* algorithms have been successfully implemented to uncover *hidden patterns* in data. This fundamental capability of clustering, and other techniques, of being able to elicit hidden patterns from data makes it a very important data mining technique and is an area of active research in machine learning, pattern recognition, and statistics, to name a few.

In data mining, the large amounts of data, and the typically large number of attributes for each instance of data, introduce some further complications in many techniques, and this is certainly the case with clustering techniques.

²⁸ For example, http://www.nhlbi.nih.gov/guidelines/obesity/bmi_tbl.htm

The computational constraints on how to deal with eliciting hidden patterns within large sets of data have resulted in many efficient algorithms being developed through such research, especially in the machine-learning discipline.

Clustering algorithms come in many flavors including:

- Hierarchical methods (agglomerative, divisive)
- Partitioning methods (relocation, probabilistic, k-means, density-based)
- Grid-based methods
- Constraint-based methods
- Machine learning (gradient descent, evolutionary)
- Scalable clustering
- High-dimensionality data

2.21.7 *Simple Statistical Measures*

We introduce a number of basic statistical measures that are widely used in data mining in the table below without significant explanation since statistical analysis is the subject of a chapter of this book (Table 2.1).

These and other statistical measures provide a generalized toolkit for data mining. In fact, statistical analysis typically plays a primary role in any data-mining effort.

2.21.8 *Linear Models*

In many cases, accurate models can be developed where the relationship between the response and explanatory variables can be assumed to be a linear function of the parameters,

$$Y = X\beta + \varepsilon, \quad (2.1)$$

where β is a set of unobservable parameters and ε is a vector of “error” values with expected value of 0 and variance σ^2 . Our objective is to then infer the values of β and ε , typically using *maximum likelihood* or *least squares*.

In cases of simple linear regression, where we have one explanatory variable and two parameters, the above equation reduces to

$$y = a + bx + \varepsilon.$$

Table 2.1 Simple statistical measures

Location (central tendency)	Mode: most common value Median: middle value Mean: $\bar{x} = \frac{\sum_{i=1}^n x_i}{n}$
Variation	Range: high–low Quartiles: Q1 (25%), Q2 (50%), Q3 (75%) Variance: $s^2 = \frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n-1}$ Standard deviation: $s = \sqrt{\frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n-1}}$ z-score: $z = \frac{x_i - \bar{x}}{s}$
Confidence levels	Mean (>30 observations): $\bar{x} \pm z_C \frac{s}{\sqrt{n}}$ Mean (<30 observations): $\bar{x} \pm t_C \frac{s}{\sqrt{n}}$ Proportion: $p \pm z_C \sqrt{\frac{p(1-p)}{n}}$
Heterogeneity	Gini index: $G = 1 - \sum_{i=1}^n p_i^2$; normalized: $G' = \frac{G}{(k-1)/k}$ Entropy: $E = -\sum_{i=1}^n p_i \log p_i$; normalized: $E' = \frac{E}{\log k}$
Asymmetry or skewness	Skew: $\gamma_1 = \frac{m_3}{\sigma^3} = \frac{\sum_{i=1}^n (y_i - \bar{y})^3}{\frac{1}{n} \sum_{i=1}^n (y_i - \bar{y})^3} = \frac{1}{n} \sum_{i=1}^n \frac{(y_i - \bar{y})^3}{\sigma^3}$ Measure of how long the tail of the distribution on one side or the other. $\gamma_1 < 0$ means the distribution skews to the left; $\gamma_1 > 0$ means a skew to the right
Kurtosis	Kurtosis: $\gamma_2 = \frac{m_4}{\sigma^4} - 3 = \frac{\sum_{i=1}^n (y_i - \bar{y})^4}{\frac{1}{n} \sum_{i=1}^n (y_i - \bar{y})^4} - 3 = \frac{1}{n} \sum_{i=1}^n \frac{(y_i - \bar{y})^4}{\sigma^4} - 3$ Measure of “peakyness” (leptokurtotic) or “flat-toppedness” (platykurtotic) of the distribution. Normal distribution has $\gamma_2 = \frac{m_4}{\sigma^4} = 3$, hence the “–3” value
Hypothesis test	1. Specify the null hypothesis (e.g., $H_0 : \mu = \mu_0$) and the alternate hypothesis (e.g., $H_a : \mu > \mu_0$) 2. Select significance level (e.g., $\alpha = 0.05$) 3. Compute the test statistic (t or z) 4. Determine the critical value for t or z using $\frac{\alpha}{2}$ for two-sided tests 5. Reject the null hypothesis if the test statistic falls into the “reject H_0 ” region
Comparing (more than 2) groups	Categorical data: chi-square test Continuous data: one-way ANOVA
Comparing variables	Correlation coefficient (r): $r = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{(n-1)s_x s_y}$

Maximum-likelihood estimation (MLE) is used to make inferences about the parameters of the underlying probability distribution for a given dataset and is very widely used across a multitude of scientific disciplines. We typically assume that the data is independent and identically distributed (iid) within a particular distribution, where the parameters are unknown. We then use MLE to create estimators for the unknown parameters.

For example, we may be interested in the weights of patients. We have a sample of our patient population, but not the complete population. We shall also assume

that the weights are normally distributed,²⁹ but with a mean and variance currently unknown. We would now use MLE to determine the mean and variance for the specific $N(\mu, \sigma^2)$ by “fixing” the data and picking the distribution parameters that are “most likely” given the data.

We have already mentioned the fact that all data contains noise: even if we manage to keep all of our independent variables constant, the dependent variables (our outcomes) will still vary, and so, as we have already indicated, we need to estimate the distribution underlying the dependent variable(s). The method of *least squares* is another method for allowing us to estimate the parameters of the distribution.

Essentially, we want to find the best fit curve for our data, we can look at the “errors” of a given set of data, given a particular curve. That is, we can look at how large the offsets (or residuals) of our data values are given a particular curve.

Given a dataset $(x_1, y_1), \dots, (x_n, y_n)$, we are interested in determining a function $f(x)$ that will be the best fit for our data. However, we assume that each point has an associated error value, $e_i = y_i - f(x_i)$, which we wish to minimize – ideally to 0. As we shall see later, we are actually looking for $f(x, a)$, where a is the set of parameters for the underlying distribution.

The Gauss-Markov theorem³⁰ proved that for a linear model where the errors have a mean of zero, with equal variances and which are uncorrelated to each other, the least squares method provides the best estimators for the underlying model.

2.21.9 Neighbors and Distance

Our intuitive definition of a *neighbor* is someone, or something, that is located near another. This simple concept is important in many areas of data mining, but especially so to the concept of *clustering*. Our concept of *neighbor* can be used to offer a measure of *similarity* between two instances or variables. During our high school education, we encountered the concept of Euclidean distance between two points. We can use this for illustration here also.³¹ Consider the example where we are capturing patient age and weight, $(a_1, w_1), \dots, (a_n, w_n)$, for our patient population, and we have the resulting data (18, 110), (25, 121), (37, 145), (49, 178), (55, 187), (63, 176), (72, 169). We can plot this data to provide Fig. 2.11.

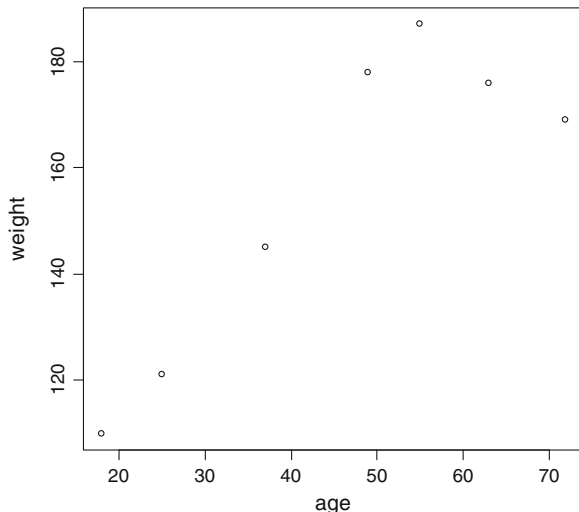
By simple visual inspection, we can see that the second data point (25, 121) is nearer to the first (18, 110) than to the third (37, 145). (Our ordering of the data points simply flows from left to right.) But it’s not quite as obvious whether the sixth data

²⁹ As we have indicated, we are interested in making inferences about parameters of the *underlying* probability distribution. We thus need to be able to define which family of probability distributions the data is a part of. We will discuss this and other issues more fully in Chap. 6.

³⁰ Proven by Carl Friedrich Gauss in 1829.

³¹ In fact, as we shall see later, the Euclidean concept of distance is used in many algorithms.

Fig. 2.11 Distance and neighborhood



point (63, 176) is closer to the fifth (55, 187), or the seventh (72, 169), but we can use a very simple equation, found widely on the Internet, to compute a distance value.³²

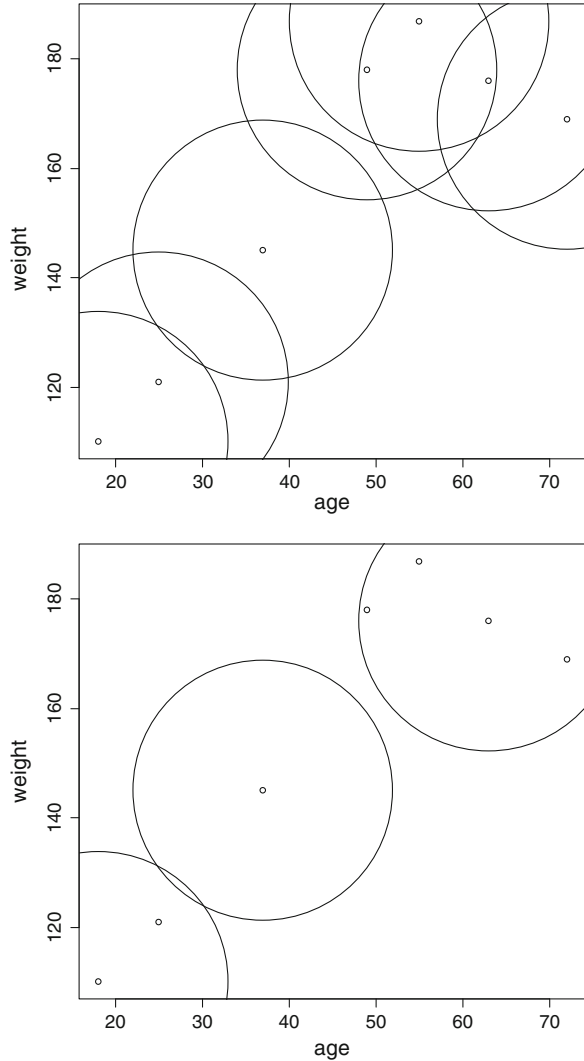
We italicized the words similarity and neighbor above. Intuitively, neighbor and distance have an important relationship, but the term neighbor needs a little more explanation and is, in many techniques, tied to the concept of clustering data points together, although not always. We'll treat these concepts much more formally, and in more detail, later in the books. For now, we'll provide a more informal discussion, illustrating with our two-dimensional data from above.

As in the real world, a neighbor is simply something that is close. Our distance measure allows us a means for determining how close two of our data points are, but it doesn't necessarily provide us with a means of determining how distant is "too distant." If the distance measure between two points A and B is 3, and the distance between A and C is 4, is the difference in distance significant? Obviously, without some context, we don't know. More often than not, the context will be relevant to the data we are studying, the models we use, the methods we are following, and the conclusions we are drawing. However, we can intuit some characteristics:

- Data points with smaller distance measures will likely be more similar to each other than those with larger distance measures (because the underlying data will have been used as input to our distance measure algorithm).
- Data points with smaller distance measures will likely be in the same neighborhood, whereas those with larger distances will more likely be in different neighborhoods (assuming our definition of a neighborhood).

³² <http://mathworld.wolfram.com/Distance.html>, for example.

Fig. 2.12 Neighborhoods



So in addition to our distance measure, the definition of the neighborhood is an integral part of our ability to accurately and effectively analyze our data. So how do we define our neighborhoods?

Simply, these will be dependent on our analysis and on your data; we will discuss this concept in detail later in the text. For now, we consider a neighborhood to be a localized area sized according to specific distance value. That is, we will consider anything with a distance value less than some limiting value d_{\max} . This will give us circular neighborhoods for 2D data, spherical for 3D, and so forth.

Figure 2.12a shows our data overlaid by circles with an arbitrary radius of 15, from which we can see that there is a way in which we can separate the data points

into three neighborhoods. Figure 2.12b depicts the same data, but with only the three neighborhoods highlighted.

We should make some observations about our separation of data into neighborhoods. The first is that we used a single parameter for our neighborhoods so that each one is the same size. This certainly simplifies some aspects of the model, but may not be pertinent. Second, we took an arbitrary size – really – which is obviously not appropriate in *any* analysis. It just worked for our illustration. But there is an important point this illustrates: what would happen if we received additional data elements that filled in some of the gaps in our data? How would, or should, our neighborhoods change? This brings us to a third point: we partitioned our data into three neighborhoods, or would have if we'd taken a slightly smaller radius and avoided the overlap between the first and second neighborhood: but how many is appropriate? The larger the number of neighborhoods, the more specific each is, and vice versa. These will also affect where new data is positioned, or how new data affects the regions previously defined.

2.22 Feature Selection

We now turn our attention to the topic of feature selection. Features and attributes are often used synonymously in the literature. We make the distinction so that features are those subsets of attributes that we, or our models, have selected and will use to make decisions. With this definition, we can see that our feature set will often be a subset of our attribute set. If we consider a specific data mining technique, this will almost always be the case, except in the most trivial of situations, since it is unlikely that we would want every attribute in our dataset to be used in any given situation. The question that we, therefore, need to ask is how do we select the appropriate attributes for the techniques and/or models we are considering? A corollary is how do we determine the *best* attribute(s) to use?

Later in this book, we discuss machine-learning techniques in detail. We will use these to illustrate an example. We will not define what a machine-learning system is, but leave the intuitive definition of a system where the learning is done by the model itself. Consider a simple machine-learning system that includes a decision tree component, such as the C4.5 system (see [Appendix B](#)). Since decision trees will select the most appropriate attribute to use to split the tree at any given point, it would seem intuitive that we should allow the system to select from as many attributes as possible. If we supply every attribute contained in our dataset, for example, then the decision tree learner *should* select the most appropriate at each point – correct?

Unfortunately, this isn't always the case. Witten (2005 #11) used an example for the C4.5 system and the effect of introducing a random binary attribute (populated by tossing an unbiased coin) into a dataset used by the system and suggested that the performance deterioration may be as much as 5–10% in the experiments performed. But why is this?

In the case of a decision tree, we are bifurcating based upon an attribute. As we drill down the tree, less and less data is available to us, and a random attribute could look better than any other attribute, and this problem increases the deeper we go down the tree.

The problem occurs with rule learning systems also, and for similar reasons: less and less data is used for the basis of decision-making, and so irrelevant attributes can have an overwhelming effect on the learning process. In general, the class of learning systems referred to as instance-based learning systems, which we describe later, typically works with what are called “local neighborhoods,” and so irrelevant attributes can have a significant impact on their accuracy and effectiveness.

We mention in passing that there are models for which this issue does not arise: the naïve Bayes model does not suffer from this problem due to its assumption (and design) that all attributes are independent. It, however, suffers in other ways, which we describe later.

In our discussion so far we have concentrated on those attributes which can be classified as irrelevant. This is not the whole story, however: relevant attributes may have a similar impact. An example from Witten (2005 #11) again suggests that relevant attributes selected at the wrong point can lead to sparse datasets being used subsequently, thus affecting the capability of the model.

To overcome these issues, we typically include an attribute selection step in our process. By weeding out those attributes that are irrelevant, and to focus on only those attributes which are most relevant, we hope to be able to optimize our model. This activity of *reducing the dimensionality* of our data can have a tremendous positive impact on our model. It can also, however, be a challenge if we eliminate the wrong attribute. Later in this book, we describe the method of *principal component analysis* (PCA) that iteratively identifies the most important components of a dataset and allows us to reduce the dimensionality in a deterministic way. Reducing the dimensionality also allows us to have a more easily understandable and interpretable definition of our problem and conclusions since we have eliminated as many of the attributes as possible and can thus focus on only the most important attributes.

We can intuitively take two approaches to selecting the features of interest, which are described in the following sections.

2.22.1 Global Feature Selection

We define *global feature selection* as selecting the attributes we are interested in solely on the basis of the dataset under study and without any consideration of the approach(es) we will use to mine the data. This is sometimes referred to as *filtering* the data prior to feeding it into our models.

One of the first challenges we encounter with this method is to determine the relevance of any attribute we are considering for exclusion. This is fundamentally

difficult because we have to understand the *relevance* of the attribute to the overall problem at hand. If we exclude consideration of how a particular method may define relevance in its learning phase, how can we be sure that we understand that attributes' relevance in the great scheme of things?

A second challenge is the one we have discussed at several points already – generalization. Since we are looking to exclude attributes, any system that separates the data out into different classifications will not be able to use those attributes. If we exclude no attribute, the system may end up separating every instance, or close to every instance, into its own partition in the solution space. If we exclude every attribute except one, the system will partition the instances into; however, many partitions map to the number of different values for that one attribute.³³ Whatever the model does during the learning (training) phase may lead to the overfitting issue we have already discussed. This global selection is, therefore, not as simple as might at first be thought. When we consider the issue of noise, the problem takes on a further nuance in that inconsistencies due to noise may result in the system overcompensating.

2.22.2 Local Feature Selection

We define *local feature selection* as selecting the attributes we are interested in with a focus on the methods we will be using to mine the data as well as the dataset itself. This is sometimes referred to as the *wrapper* method: the model(s) for mining is wrapped into the selection process. We should also mention at this stage that we might use methods that will *not* be used directly in our mining efforts to select features for methods which *will* be used. An example where this is very valuable to us is when we intend to use *nearest neighbor* methods. We will discuss these in more detail later in the book but introduce one characteristic of these models that makes them challenging to use. They are very susceptible to being biased by irrelevant attributes. To overcome this, other models, such as decision trees, can be used to remove irrelevant attributes and thus improve the performance of the nearest neighbor algorithm. Another approach we can use is to build a linear model and rank the attribute coefficients.

For the interested reader, Puuronen and Tsymbal (2001 #1035) has an interesting discussion on local feature selection in the context of classification, a topic we will return to later in this text. Ribeiro et al. (2010 #1037) considered this topic with specific emphasis on text clustering. Yijun (2010 #1038) described this challenge in the context of high-dimensional data analysis.

³³ We recognize that this is somewhat of a simplification because the classification method may still classify unique instances into the same class or may result in a smaller number of classifications than the number of discrete values for the single attribute.

2.23 Conclusion

We intentionally covered a wide range of topics in this chapter, many of which we shall return to in later chapters so that we can cover them in more detail. Our purpose in this chapter was to broadly introduce the wide range of challenges and opportunities which have to be considered in a data mining effort and to discuss various approaches we can take to mine our data. We can very quickly, and simply, begin to get a sense of our dataset by using a simple method such as 1R, or to begin looking for simple association rules, for example. In so doing, we can often see an effective path to pursue in order to uncover more and more value; an iterative approach that will serve us well.

How our dataset is represented can severely compromise which approaches we can use and how effective they are. Missing data and noise are two significant issues and need to be “eliminated” before we move forward. We have to be careful that our overt and subconscious decisions don’t result in a model that is overfitted to the data we use for the learning stage and thus has reduced generalization. Selecting the attributes (features) which are most relevant, and defining relevance itself, is not a trivial exercise, but can be relieved by using some models to help.

A significant part of the effort involved in data mining actually occurs *before* we even get to the analytics phase: preparing our data for analysis by getting it in the correct format, cleaning up the noise and errors, and providing data that the techniques we choose to leverage can actually use should not be minimized. Effort we put in at this stage will save us pain and heartache later on.

We will build upon the concepts we have introduced in this chapter as we explore the various issues that arise from the underlying data architecture and how we might need to restructure it in order to more effectively elicit the hidden patterns we are striving to understand; the input and the transformations necessary to convert it from generated data to analyzable data; how we wish to represent the output for our users, customers, and downstream systems; and what are the most appropriate techniques for us to use as we leverage statistical and machine-learning methods, among others, to support our data mining goals.

References

- Bellman RE (1961) Adaptive control processes: a guided tour. Princeton University Press, Princeton
- Duda RO, Hart PE, Stork DG (2001) Pattern classification. Wiley, New York
- Gallier JH (1986) Logic for computer science: foundations of automatic theorem proving, Harper & Row computer science and technology series. Harper & Row, New York
- Graner F, Glazier JA (1992) Simulation of biological cell sorting using a two-dimensional extended Potts model. Phys Rev Lett 69:2014–2016
- Han J, Kamber M (2006) Data mining: concepts and techniques. Morgan Kaufmann, San Francisco
- Kohavi R (2000) Data mining and visualization. US Frontiers of Engineering

- Puuronen S, Tsymbal A (2001) Local feature selection with dynamic integration of classifiers. *Fundam Inform* 47:91–117
- Rabin MO (1980) Probabilistic algorithm for testing primality. *J Number Theory* 12:128–138
- Ribeiro MN et al (2010) Local feature selection for generation of ensembles in text clustering. *Neural Networks (SBRN), 2010 Eleventh Brazilian Symposium on*. pp 67–72
- Tietz NW (1995) *Clinical guide to laboratory tests*. W.B. Saunders Co., Philadelphia, pp xxxix, 1096
- Tufte ER (1990) *Envisioning information*. Graphics Press, Cheshire
- Tufte ER (1997) *Visual explanations: images and quantities, evidence and narrative*. Graphics Press, Cheshire
- Tufte ER (2001) *The visual display of quantitative information*. Graphics Press, Cheshire
- Witten IH, Frank E (2005) *Data mining: practical machine learning tools and techniques*. Morgan Kaufman, Amsterdam/Boston
- Yijun S (2010) Local-learning-based feature selection for high-dimensional data analysis. *IEEE Trans Pattern Anal Mach Intell* 32:1610–1626



<http://www.springer.com/978-1-58829-942-0>

Introduction to Data Mining for the Life Sciences

Sullivan, R.

2012, XVIII, 638 p., Hardcover

ISBN: 978-1-58829-942-0

A product of Humana Press