Chapter 2

Modelling

2.1 Basic concepts

The basic concept of modelling is a *model* – an abstract representation of a real world process. For example, scientific theories are models of reality, which try to explain it and predict future phenomena. In computer science, models are defined more tightly. In the following, we define the basic concepts used in this theses:

A **model** $\mathcal{M} = (S, \theta)$ consists of a model structure S and assigned parameter values θ. I.e. it is an instance of a given model structure.

The **model structure** S is a structure which determines the parameters required for constructing a model. Typically the structure consists of variables and their relations (e.g. conditional dependencies). A less obvious structure is e.g. a linear regression equation $\hat{Y} = \alpha + \beta_1 X_1 + \beta_2 X_2$, which determines how the predicted variable Y depends on explanatory variables X_1 and X_2, but does not fix the parameters α, β_1, β_2.

The **model parameters** θ are assigned numerical values like probabilities in Bayesian network or linear coefficients (above α, β, γ) in linear regression.

When we want to find a model describing the problem domain, we should first decide the **modelling paradigm** – the general modelling principles used. The modelling paradigm consist of basic definitions, assumptions, and techniques for constructing and using certain kind of models.

The next step is to define the model structure. The search space is often called a **model family** – a set of possible model structures, for example all Bayesian networks with 3 nodes.

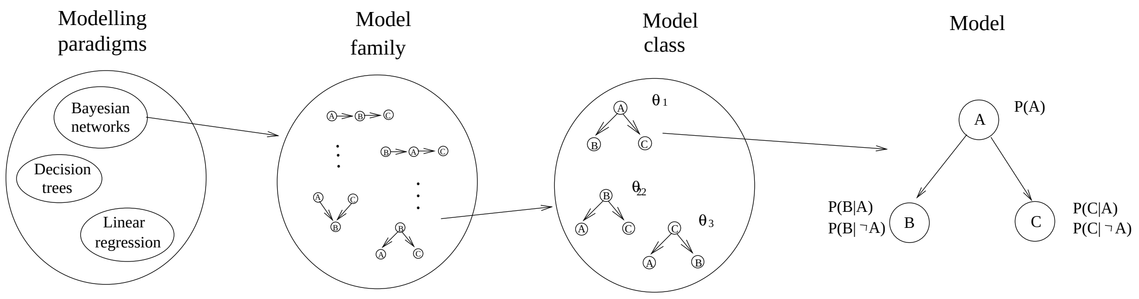

Figure 2.1: Modelling paradigms like Bayesian networks, decision trees and linear regression describe the general modelling principles used. A model family is a set of model structures in the given modelling paradigm. E.g. in Bayesian networks modelling paradigm a model family consists of different graph structures with variable nodes and conditional dependencies (edges) between variables. A model class is a set of models with a fixed structure but different parameters θ_i. In the case of Bayesian networks the parameters are prior probabilities associated to root nodes and conditional probabilities associated to non-root nodes. A model is an item in a model class with a fixed structure and fixed parameters.

When the model structure is fixed, we still have several possible models with different parameter settings. This set is called a **model class**. A model class is a set of models with the same structure and they can be considered as instances of a subset in model family.

The hierarchy of modelling concepts with some examples is presented in Figure 2.1.

Another concept often used in data mining is a **pattern**. A pattern is a local model, which describes only part of problem domain. For example associative rules describe only some dependencies between variables in the data, while a Bayesian network describes all dependencies (conditional probabilities) between variables.

The models can be further classified as *global* or *local*. The local models are usually called *patterns*, and concept "model" is reserved only for global models. A global model describes the whole data set and/or is able to make predictions for any data point. For example, all students in *Programming* course can be classified as beginners, novices or experts according to the prequisite test. On the other hand, a pattern describes only a relatively small part of data, and thus they are quite unsuitable for prediction. For example, most students who are good in *Data Structures* course are typically good also in *Theory of Computation*.

2.2 Descriptive and predictive modelling

The main tasks of knowledge discovery can be classified as *descriptive* and *predictive modelling.*

Descriptive modelling is the core of classical data mining. In descripitive modelling we describe the data or its producing process and try to give *explanations.* We search for groupings and dependencies which ”naturally” hold in the data. Usually the goal is to find only simple models or local patterns.

The typical tasks of descriptive modelling are:

- Clustering: identifying a finite set of categories or clusters to describe data.
- Summarization: finding a compact description for a set of data. The description can be for example a probability distribution.
- Dependency modelling: finding a local model (a pattern) that describes significant dependencies between variables in the given data set.
- Change and deviation detection: discovering most significant changes in the data set. An important application is to detect *outliers*, untypical datapoints, which do not fit the general model.

For example, in educational context, we can search dependencies between students' performance in Programming and Data structures courses, or cluster the students into five clusters according to their weekly exercise points. Sometimes also *information retrieval* (retrieving the relevant documents for the user based on their content) and *explatory data analysis* (*EDA*) (interactive and visual techniques to represent the data visually for the human interpreter) are included to descriptive descriptive tasks of KDD.

In *ViSCos* project, most of the use cases concern descriptive modelling. We would like to find common features for drop-outs, failed and really good students, as well as identify tasks which indicate good or bad success. In addition, we would like to find groupings of students and tasks and relations between those groupings.

Predictive modelling is the core of classical machine learning. In predictive modelling we search the most plausible model which has produced the data, and which can be used to predict the future outcomes. The goal is to find a global model, which can be quite complex. The typical tasks of predictive modelling are:

- Classification: discovering a function, which classifies the data into predefined classes.
- Regression: discovering a function, which maps a data item to a real-valued variable.
- Bayesian modelling: learning a probabilistic model, which describes the dependencies between attributes, and can be used for predicting reasons and effects.

For example, we can predict the student's success in the course, given the previous study record, or diagnosize the reasons for good or poor success. These are also the most important use cases in the *ViSCoS* project: to predict the drop-outs, failed students and possibly also good students as early as possible.

According to our view (Figure 2.2), descriptive and predictive tasks are complementary phases of the same modelling process. This view is especially useful in adaptive learning environments, in which the model can be developed throuh several courses. The existing data is analyzed in descriptive phase, and a desirable model class is defined. An initial model is learnt, and applied to new data in prediction phase. After the course, the new data is analyzed, and the old model is updated or a new better model contructed.

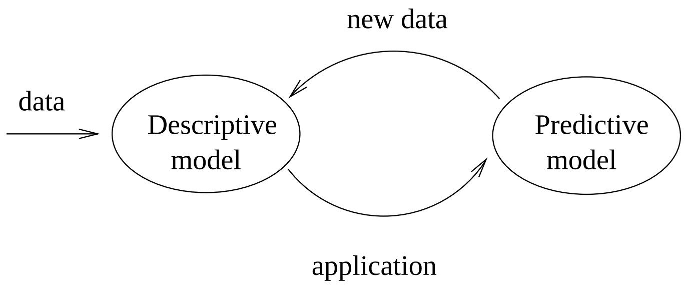

Figure 2.2: Iterative process of descriptive and predictive modelling tasks. Descriptive models reveal underlying patterns in the data and guide the selection of the most appropriate predictive model. When the predictive model is applied in practice, we gather data for the new descriptive models. As a result, the next generation predictive models are improved in every cycle.

2.3 Target functions

The goal of data modelling can be defined exactly by *target functions.* This concept is especially used in classification literature instead of models. Mathematically, functions are restricted type of models, but with certain relaxations we can describe all models as functions.

The underlying assumption is that in principle we could model the reality perfectly, i.e. the attribute in interest is a function of other attributes. Formally, a target function is any function defined over data set, which gets a set of known attributes X as its parameters and calculates the value of some new attribute Y, which we are interested. In an ideal case, we can efficiently find a function $f : X \rightarrow Y$ such that $f(\pi_X(t)) = \pi_Y(t)$ for all tuples $t \in r$. The form of function (model structure) can be given, like in linear regression, where we try learn linear functions of form $Y = \alpha + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_k X_k$ for predefined $X_1, ..., X_k$ and Y. Sometimes, even the function form should be learnt, and we know only what kind of function we are looking for. For example, in decision trees, we want to learn a set of boolean-valued expressions (composed features) $F_i(t)$, for which $F(t) = 1$, only if t's class is c_i. This corresponds to situation, when we have fixed only the modelling paradigm, but not the model structure.

In practice, we can seldom learn the target function exactly or it can be computationally unfeasible, and we have to content ourselves with its approximation. That is why the learnt function (model) is often called a *hypothesis.* In learning phase the best hypothesis is selected according to some *score function*, and the new observations can either confirm or weaken the hypothesis. In educational applications, it is also typical that the data set is itself *inconsistent*, i.e. for some tuples t_1 and t_2 $t_1[X] = t_2[X]$, but $t_1[Y] \neq t_2[Y]$. In that case, we cannot define any function $f : X \rightarrow Y$ for all $t \in r$. It is possible that we could find a function in higher dimensions (containing new attributes X_i), but with the current data we can learn only partial functions.

The target functions are explicitely used only in predictive modelling, but the type of target functions can also demonstrate the difference between predictive and descriptive modelling. In both approaches, we are searching functions of some given form or type. In predictive modelling, we have fixed the input attributes (function parameters) and the output attribute and try to find a function, which best approximates the output attribute value for **all** data points given the input attribute values. Because the function covers all data points, it is called global. In addition, we have usually already selected the maximum set of potentially relevant input attributes $X_1, ..., X_k$ and the function uses all k attributes as its parameters.

As a contrast, in descriptive modelling, we usually have not fixed the input and output variables, but try to find all functions, which approximately hold for X and Y. The resulting functions are often defined only for a projection of r onto some attribute set $X \subseteq R$. In addition, the function does not have to hold for all datapoints, but the domain can be restricted to some subset $r' \subseteq r$. That is why the functions are called *partial*. They do not correspond to any global model, but a local pattern. In mathematics, this corresponds to a resctriction of function $f|_r'(t)$. In practice this means that we should discover both the function (of some restricted type) and the subset $r' \subseteq r$, where function is defined. For example, associative rules describe functional dependencies between some attribute-value pairs $X = x \Rightarrow Y = y$, which hold for some large enough subset of r'. The resulting functions are characteristic functions, with restricted domain $\sigma_{X=x}(r)$. In addition, we are usually interested in only the tuples, for which the relation holds, i.e. $f|_{X=x}(t) = 1$.

Clustering is an example of descriptive modelling, in which we are searching a global target function. The goal is to find a function approximation $f : r \rightarrow \{c_1, ..., c_m\}$, which maps each data point to its "best" cluster c_i. Unlike in predictive modelling, the goal set $\{c_1, ..., c_m\}$ is not given, but we should also decide the number of clusters and find the best clusters.

2.4 Model selection

The principal problem in data modelling is *model selection*, i.e. how to select the best model, for the given data and problem. The problem can be divided into three subproblems:

1. How to select the best modelling paradigm?
2. How to select the best model structure and thus the model family?
3. How to select the best model in a given model family?

Usually the most emphasis has been put on steps 2 and 3, which can be solved by suitable score functions. However, step 1 already rules out a vast number of possible models, and it is possible that the data cannot even be modelled by the selected paradigm. Analogically, the paradigm is like a grammar, which defines what we can express in the language. We cannot express Russian sentences by Finnish grammar, even if the words are correct. Unfortunately, the underlying assumptions in paradigms are often implicit, and the only choice is to try several paradigms and then select the best one.

Selection of model structure and model parameters are relatively well-studied problems. As we have already remarked, we can very seldom learn the exact model. Thus, we need a policy to select the best hypotheses among all alternatives. The same method of *score functions* can be applied to both problems, although trying all possible model structures can be intractable and some heuristics are needed. In such heuristics we once again commit ourselves to some assumptions, which should be made explicit, before we can judge their suitability to the given data and problem. In descriptive modelling, the model structures are relatively simple, and we can search through the whole model space. Fortunately, the score functions can also prune the search space remarkably.

In the following, we will first discuss about the model complexity, which affects on the selection of paradigm and model structure. Then we will continue on heuristics and implicit assumptions, which are used to restrict model family or select the model structure. Finally, we will discuss about the score functions used in predictive and descriptive modelling.

2.4.1 Model complexity

The desired model complexity is an important choice in model structure selection. On one hand, we want to get as expressive and well-fitting model as possible, but on the otherhand, the more spcialized model is, the more training data is needed to guarantee that the model does not overfit. *Overfitting* is the main danger of complex models in combination with small data sets like in educational domain. We say that model M overfits data, if there exists another model M' such that M has smaller error in the training set than M', but M' has smaller true error than M. I.e. the model adapts the training data so well that it models even the noise in data and does not generalize to any other data sets. Overfitting can happen even with noise-free data, if it does not represent the whole population. For example, if some test is voluntary, only the most active students tend to do it, and we cannot generalize the results to other students.

The best way to avoid over-fitting is to use a lot of data. A small training set is more probably biassed, i.e. it does not represent the real distribution. For example, the attributes can have strong correlations in the sample, even if they are uncorrelated. When large amounts of data are not available, the best is to select simple models. In practice, the attribute set can be reduced by selecting and combining original attributes. This is also an important option when selecting the modelling paradigm. Descriptive modelling paradigms impose usually very simple patterns, but in predictive modelling, the paradigm may require too complex models. For example, decision trees require much more data to work than Naive Bayes classifier. We will return to this topic in chapter **??**.

Some learning methods have preference for simplicity in their score function. The most famous such principle is *Occam's Razor*, which states that we should favour simpler hypothesis. An application of Occam's Razor in machine learning is *minimum description length principle* (*MDL*), which favours models which can be described by shortest code. However, it should be noticed that the description length depends on the representation, and MDL does not give an absolute measure for simplicity. In addition, several heuristic overfitting avoidance methods have been developed, with varying success. In some domains they can achieve great improvements, while in others they can lead to even poorer performance.

Model validation e.g. by cross-validation is especially crucial if overfitting is suspected. The goal of cross-validation is to measure, how well the model generalizes to new data points outside the training set. Some methods use the cross-validation already in the learning phase, to decide appropriate model complexity. We will return to model validation in the next section.

On the other hand, a simple model generalizes well, but it can be also too simple. Too simple model does not catch any essential features. There is also danger that if you try enough many simple patters, at least some of them will fit, even if there are no patterns at all. For example, consider a data set of totally independent binary-valued attributes, which follow binomial distribution with parameter $p = 0.5$ (i.e. each attribute has an equal distribution). If the data size is 100 tuples, any associative rule of two attributes has 35% probability to occur. This is quite high frequency, and there is a big temptation to accept such a spurious rule, if no further validation is performed. This phenomenon is typical to descriptive modelling, and historically, it has been one of the main reasons to critisize data mining. As Sullivan et al. [**?**] state it: "*Data snooping occurs when a given set of data is used more than once for the purposes of inference or model selection. When such data reuse occurs, there is always the possibility that any satisfactory results may simply be due to chance rather than to any merit inherent in the method yielding the results*".

As a compromise, we should select simple models, which still explain the data well. Very often, this happens through trial and error, in an iterative process of descriptive and predictive modelling.

2.4.2 The bias in learning styles

Every teacher knows that the learning style affects the learning outcomes, too. The same holds for machine learning. Even if the learners are committed to the same modelling paradigm and try to optimize the same score function, they can produce different models according to their learning styles. The main reason is that the learners do not perform an exhaustive search through the model space, but use some heuristic principles to prune the search space. These assumptions cause bias in the resulting model: certain kind of models are preferred to others.

The main division of learning styles is to *inductive* and *deductive learning*. The methods are same than in general scientific paradigm: in induction we infer a hypotheses from a set of examples, while in deduction the theory is applied to new instances. In pedagogical terms, inductive learner is like a student, who practices mechanically with as many examples as possible before goint to the exam, while a deductive learner tries to understand the new material in accordance with her/his previous knowledge.

In machine learning, the inductive learning is much more popular, because it enables generalization to new instances without any a prior knowledge about the domain. In deductive learning we do not in fact learn anything new, but just infer new consequences from the existing domain theory. We can for example infer new features from the data or learn how the theory is manifested in data. According to Mitchell [Mit97], deductive learning is best used as a complement to inductive learning, where it can be used to initialize the model, define the score function or guide the search. In this way, we can manage with smaller training sets, and the assumptions of the deductive theory are explicitely declared.

Inductive learning requires more training examples to work, and what is even more critical, it has to make some implicit assumptions about the data. These assumptions are known as *inductive bias* or *inductive principles*. These principles are necessary for inductive learning, because otherwise the model cannot generalize outside the training set. Often, the inductive bias is also necessary for practical reasons, because complete search in large model space is intractable.

Inductive bias has a critical role in data modelling, because it tells, which model family and algorithm works best for the given data and purpose. When the goal is predictive modelling, we can first perform descriptive analysis to discover the nature of data, and then select the methods with most suitable inductive principles. In descriptive modelling, the purpose affects on selection, for example the principle how we define a cluster.

The inductive bias can be further divided to two types [Mit97]:

1. In *restriction bias* we restrict the search space (model family). This kind of bias is also called *representational* or *syntactic*, because it restricts what kind of models we can represent. The restriction can be inherent already in the modelling paradigm. For example, linear regression model supposes that the target function is linear, and rule out all other functions. Inside a modelling paradigm, we can restrict the search space by selecting the number of attributes (model size). The bias can also concern the learning style. In *consistent learning* we suppose that the true model is fully representable in the selected model family and thus the model should have 0 training error. The stricter the restrictions, the more efficiently we can learn the model – if the reduced search space contains the true model any more.

2. In *preference bias* we do not rule our any models, but instead put preferences on them. For example, we can prefer more special models to general ones, or favour the simplest models like in Occam Razor principle. The preferences can also concern the ways to estimate parameters. Preference biasses are frequently used in heuristic optimization algorithms to navigate in the search space and find good models efficiently. For example, in greedy heuristic we always move to locally best direction, and the resulting model is not necessarily the globally optimal.

As we observe, the restriction bias concerns selection of model structure, while preference bias can concern both model structure and parameters. Preference bias is usually more recommendable, but sometimes we cannot avoid restriction bias. The effects of bias are very domain-specific, and the same bias can be either beneficial or undesirable, depending on the domain. In fact, prior knowledge about the domain can work as an efficient bias. This kind of *semantic bias* can be expressed in the form of either restriction or preference bias. This idea is occupied in our general paradigm of descriptive and predictive modelling: the results of descriptive modelling phase guide the predictive modelling.

Some of the biasses are explicitely expressed in the method and the used can manipulate them. For example, in probabilistic clustering, the user can decide the form of distribution. But more often the biasses are implicit and unchangable, and the user should just be aware of them. A typical inductive bias in maching learning is an assumption that the training set represents the whole population and thus the distribution is also same. If the future examples have very different distribution, the method does not work. A more restrictive inductive bias is the assumption of Normal distribution, which in practice holds very seldom. If the training set is very large, the error is small, but for small data sets assumption of binomial distribution is better. In clustering, there is a vast number of different methods with different biasses. These will be discussed later in Chapter **??**.

Inductive learning can be further divided into *eager* and *lazy* or *instance-based learning*. An eager learner constructs one global model during the training and the model cannot be change afterwards. In pedagocial terms eager learner is like a student, who already thinks how to apply the theory to new problems, before it is actually asked. On the other hand, a lazy learner tries to delay learning as long as possible. The learner stores all training examples, and approximates the function value for a new instance locally, according to previous examples. Thus, lazy leaner is like s student, who collects all given material, but does not try to construct any general principles. When given a problem, s/he tries to apply most similar old examples and interpolates the answer. The most common variation of lazy learning is *k-nearest neighbours* method. For every new data point the method calculates mean, median or some other function of k nearest neighbours's target function values.

In eager learning, the inductive bias is the assumption that a global model can be learnt from the training example. In lazy learning, the inductive assumption is that the new point is similar to the closest previous examples. Eager learner takes more time in the learning phase, but the application is very fast. As a contrast, in lazy learning the training is fast, but application is more worksome. In addition, lazy learning requires a lot of training data to work. That is why eager learning, or combination of both, are by default better cadnidates for educational applications.

2.4.3 Score functions in supervised and unsupervised learning

The score function measures, how well the model or pattern fits the data. An ideal measure function would reflect precisely the utility of a particular model. However, such functions are hard to define, and instead we use generic measure functions like probability, sum of squared errors, or misclassification rate. One important criterion is that the measure function should be robust, i.e. insensitive to small changes in the data.

According to machine learning terminology, predictive and descriptive modelling can be seen as *supervised* amd *unsupervised learning*. The main difference lies in the use of score functions: In supervised learning (classical machine learning) the goal is to determine the model structure and parameter values that maximize the score function, given some example data (*training set*). Because the correct answers are known, we can simply compare the predicted values to real values. In unsupervised learning, the correct answers are not known, and we should use other kind of score functions, which measure the general goodness of a model or pattern.

In pedagogical terms, supervised learning could be compared to ”teacher-centered pedagogy". We suppose a "teacher", who is teaching the system and evaluating its performance. The teacher can be a fittness function or some other external method for estimating the proposed model by learning machine. The teacher has example data – set of input values with their correct output values – and knowledge about environment (unknown to the learning system). The system tries to learn a model, which produces the most correct output values. Typically this learning happens through "trial and error": The system is given input values, and it produces output values according to its current model candidate. The performance is evaluated by the teacher according to prediction error, i.e. the difference between real and proposed output values. In addition, there may some other evaluation criteria based on teacher's preferences or knowledge about environment. For example, the teacher may favour simpler solutions to more complex ones according to *Occam's Razor principle.*

Unsupervised learning could be compared to "student-centered pedagogy". No teacher is supposed, but the learning system has to construct and evaluate the model itself. The system is given only set of sample inputs without any "correct" output values. The system has only a general goodness function to evaluate its own performance, based on apriori knowledge about the environment. For example, the score function measures, how well the pattern covers data (e.g. the frequency of an associative rule, or data likelihood given a clustering) or how informative it is (e.g. *information gain measure*). This kind of goodness function could be called a tutor, who supervises general learning process and knows some general learning principles, but does not know the correct answers on spesific tasks!

2.4.4 Overfitting avoidance and bias in *ViSCoS* project

In *ViSCoS* project, the data sets are small like in most educational applications, and the risk for overfitting is apparent. Thus, restriction to or preference for simple models is a natural bias. This preference for simplicity concerns both modelling paradigms and model sructures. For classifications tasks – predicting success or failure in the course – linear regression and Naive Bayes model are good paradigm candidates. In both paradigms, the model parameters can be defined quite accurately from a small set of data. The number of attributes is also critical for model simplicity, but in our case it is not a problem, because we have only a few attributes available. However, the attribute domains are too large and may need some reduction. This will be returned in the next chapter.

The prior knowledge on the domain would be especially valuable bias, which could compensate also the missing data. The course teachers can give some subjective knowledge, but the better approach is to discover such knowledge. This is exactly the goal of descriptive modelling. In the descriptive phase, we will use all available data to find knowledge in the form of patterns: relations like correlations and associative rules and groupings by several clustering methods. These patterns give important information about the most relevant attributes and their relations.

In clustering, the bias is especially critical, because it determines, what kind of groupings we will find. We should define the desired number of clusters and what kind of clusters are preferred. In probabilistic clustering an important question is the shape of distribution. In the grade distribution we can usually accept the Normal Distribution, but does it hold for exercise task points or difficulty of tasks? This topic will be returned in Chapter **??**.

2.5 Model validation

The aim of model validation is to give insurance that we have found a good model, or at least that we do not accept a poor model. The same techniques can also be used for comparing alternative models and select the best one. However, it is good to remember that the methods are just insurance policies, and they do not eliminate chance and variability in data. In the following, we will briefly introduce the most common validation techniques for decsriptive and predictive modelling. The tests will be demonstrated in practice in the susequent chapters.

2.5.1 Statistical tests

In descriptive modelling, we would like to get insurance that the found patterns are meaningful and not only due to chance. This is a real danger especially with small data sets, because we perform an exhaustive search for quite simple pattern structures. It is quite probable that some of the discoveries are spurious and could have occured in totally random data, as well. It is always good to follow Smyth's [Smy01] instruction and imagine, how the method would have worked with totally random data. This is exactly the goal of statistical significance tests.

The statistical tests follow the general schema of proof by antithesis. If we want to validate hypothesis H, given observation $X = x$, we make an antihesis (*0-hypothesis*) $H_0 = \neg H$. If the probability of obervation given the 0-hypothesis is very small, we reject it and accept the original hypothesis H.

In *significance test* we evaluate some test measure X and calculate the probability

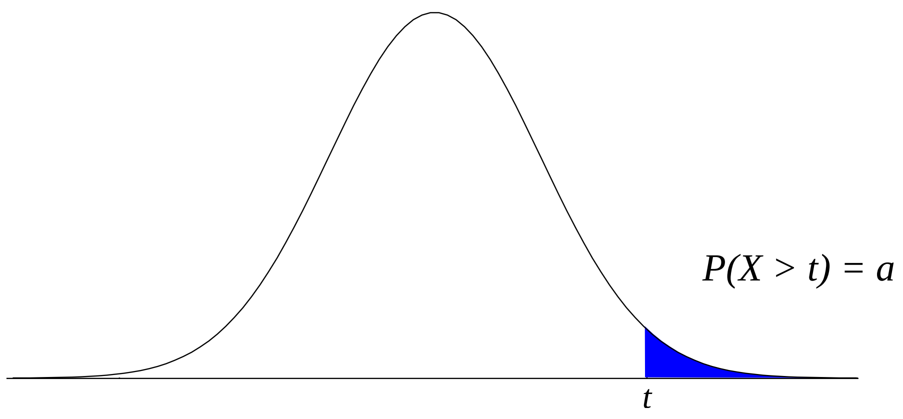

Figure 2.3: The idea of significance test. The distributional form depends on the given test measure. Value $P(X > t) = a$ is the significance level, typically 0.01, and t is the corresponding critical value, given the test distribution. If the value of the test measure $X = x$ lies on shaded area, the observation can be considered as statistically significant at the given level a.

$P(X > x) = p$[1] that observed value $X = x$ could occur by chance. If this probability is very small, we can accept the hypotheses at the *level of significance* p. The typical levels of significance are:

0.05	nearly significant
0.01	significant
0.001	very significant

The corresponding test measure values $t_{0.05}$, $t_{0.01}$,$t_{0.001}$ are called *critical levels.* Usually the critical levels are available and it is enough to compare the test measure to critical levels of the given test distribution.

χ^2-test

χ^2-test is a popular significance test used to compare distributions (variances). The uderlying assumption is that the data is approximately normal-distributed and thus the squared deviation from the expected distribution approaches χ^2-distribution. Let's suppose that we have observed interesting distribution in attribute $X \in R$.

[1]This so called *right-tail test* is used, when we have observed larger test measure value than expected. If we have observed smaller value, *left-tail test* $P(X < -x)$ is used instead. In the *two-tailed test* we calculate probability $P(X < -x \vee X > x)$.

We would like to know, if this phenomenon is just due to chance or if we have found something extraordinary. If X can have m different values $x_1, .., x_m$, we calculate test measure

$$\chi^2 = \Sigma_{i=1}{}^m \frac{(m(x_i) - e(x_i))^2}{e(x_i)},$$

in which $e(x_i)$ the expected value of $m(x_i)$. Expected frequencies are evaluated under assumption of random data, and thus $e(x_i) = n/m$ for all i. If the measure is greater than the critical value $t_p(m-1)$ for selected level p and $m-1$ degrees of freedom, then the we can suppose that with p probability the observation is not due to chance.

If the observed frequencies are very small, the the measure does not follow χ^2-distribution and test becomes less accurate. As a rule of thumb, all frequencies should be at least 5. If this this does not hold, we can try to combine or redefine attribute values or use binomial test instead.

Binomial test

Binomial test is a better alternative than χ^2-test, when the frequencies are small, which is often the case with educational data. Now the observed attribute can have only two values or it should be made binary-valued. For example, if X can have m values, we can still study cases $X = x$ and $X \neq x$. Like in previous example, we suppose that in random data all values have equal distribution, and thus $p = P(X = x) = 1/m$ and $q = P(X \neq x) = \frac{m-1}{n}$. Now we can calculate the probability that our observation $m(X = x) = k$ is significant. If we suppose random data, the observation holds with probability

$$P(m(X = x) \geq k) = \Sigma_{i=k}{}^n \binom{n}{i} p^i q^{n-i}.$$

If this probability is less than 0.01, we can assume that discovery is significant.

2.5.2 Independence test

With independence test we can measure, whether the dependency between attributes is significant. Let us consider the simple case, when we have two binary-valued attributes A and B. Now the 0-hypothesis is that the A and B are independent, and thus $P(A, B) = P(A)P(B)$. Otherwise we proceed as in χ^2-test, but now

Table 2.1: 2×2 contingency table.

	B	$\neg B$	Σ
A	$m(A, B)$	$m(A, \neg B)$	$m(A)$
$\neg A$	$m(\neq A, B)$	$m(\neq A, B)$	$m(\neg A)$
Σ	$m(B)$	$m(\neg B)$	n

the expected frequencies are calculated under independence assumption from $m(A)$ and $m(B)$. We get the counts from 2×2 contingency table (Table 2.1).

Now the test measure

$\Sigma_{i=0}{}^{1}\Sigma_{j=0}{}^{1}\frac{(m(A=i,B=j)-e(A=i,B=j))^2}{e(A=i,B=j)}$

follows χ^2-distribution with 1 degree of freedom.

The test can be easily generalized for non-binary attributes. The only difference is the number degrees of freedom, which is $(r-1)(c-1)$, if $|dom(A)| = r$ and $|dom(B)| = c$. Once again, if the frequencies are very small, the test cannot be used and we should use *Fisher's exact test*, instead.

2.5.3 Testing prediction accuracy

In predictive modelling, we have slightly different aims. We would like to ensure that the model has not overfitted and generalizes well. Small training error does not give any guarantees about good prediction accuracy in the future, because we can achieve zero training error for any data set, if we just select sufficiently complex model. As a solution, different kind of testing schemas are used.

In the ideal case, we can reserve part of data as a validation set and use the rest for training. Now the validation set gives good outlines, how well the model works with unseen data. However, if the original data set is very small, this only increases the risk of overfitting. In this case, *(k-fold) cross-validation* is a better solution. The idea is that we partition the original data set to k disjoint subsets of size n/k. Then we reserve one subset for validation and learn the model with other $k-1$ subsets. The procedure is repeated k times with different validation set and finally we calculate the mean of prediction errors. The same method can be used for comparing different models.

The most popular error functions for measuring prediction error are *Sum of squared errors* (*SSE*) and *classification rates*. *SSE* is defined as

$$SSE = \Sigma_{i=1}{}^{n}(y_i - f(\overline{x_i}))^2,$$

in which y_i is the real value $f(\overline{x_i})$ the predicted value of data point $\overline{x_i}$ $(i = 1, ...n)$. *SSE* can be used to measure prediction error of any numeric-valued target function f. In addition, it is often used as a score function in learning phase.

In classification, where the predicted values are typically categorial, *classification rates* are often used. Let us first consider the case of two classes, c_1 and c_2:

true positive $m(f(\overline{x} \in c_1 \wedge y \in c_1)$	false negative $m(f(\overline{x} \in c_2 \wedge y \in c)$	$\Sigma =$ $m(y \in c_1)$
false positive $m(f(\overline{x} \in c_1 \wedge y \in c_2)$	true negative $m(f(\overline{x} \in c_2 \wedge y \in c_2)$	$\Sigma =$ $m(y \in c_2)$
$\Sigma = m(f(\overline{x}) \in c_1)$	$\Sigma = m(f(\overline{x}) \in c_2)$	

Now the rate

$$\frac{\text{true positive} + \text{true negative}}{n}$$

tells the classification accuracy (proportion of correctly classified examples), and

$$\frac{\text{false positive} + \text{false negative}}{n}$$

tells the classification error. When we have more classes, we simply calculate true positive and false negatice for all classes.

2.6 Further reading

A comprehensive description of different learning styles can be found in [Mit97]. We have skipped *genetic algorithms* [Hol62, Hol92], because they are more an optimization method than a model of their own. However, genetic algorithms do also restrict what kind of models we can learn – i.e. they contain their own bias. In the same way other optimization techniques like simulated annealing [KGV83] and tabu search [Glo86] can be used in knowledge discovery. In instance-based or lazy learning, we could also mention *case-based reasoning* (e.g. [Kol93]), which is quite similar than k-nearest neighbours method, but works on symbolic data.

The bias in modelling paradigms and learning styles is unfortunately quite neglected topic in literature. [Mit97] has done a pioneering work by formalizing the inductive bias and analyzing the bias in different machine learning algorithms. The idea of overfitting avoidance as a good or bad bias is discussed in [DHS00] and [Sch93]. [EC] has risen discussion about inductive bias in clustering algorithms. In this theses, we have processed the topic further.

Our introduction to statistical tests have been very brief, and we recommend further reading. As [Smy01] states it "*Data mining algorithms should not be a substitute for statistical common sense*". [MA03] is an easy-reading and quite comprehensive textbook on probability and statistics. For Finnish readers, we also recommend [Kar01].