

<https://helda.helsinki.fi>

Classifiers for educational technology

Hämäläinen, Wilhelmiina

CRC press
2010

Hämäläinen , W & Vinni , M 2010 , Classifiers for educational technology . in C Romero , S Ventura , M Pechenizkiy & R S J D Baker (eds) , Handbook on Educational Data Mining . Chapman & Hall/CRC Data Mining and Knowledge Discovery Series , CRC press , pp. 54-74 . < <http://www.crcpress.com/product/isbn/9781439804575> >

<http://hdl.handle.net/10138/24369>

submittedVersion

Downloaded from Helda, University of Helsinki institutional repository.

This is an electronic reprint of the original article.

This reprint may differ from the original in pagination and typographic detail.

Please cite the original version.

Classifiers for educational data mining

W. Hämmäläinen and M. Vinni

1 Introduction

The idea of classification is to place an object into one class or category, based on its other characteristics. In education, teachers and instructors are all the time classifying their students for their knowledge, motivation, and behaviour. Assessing exam answers is also a classification task, where a mark is determined according to certain evaluation criteria.

Automatic classification is an inevitable part of intelligent tutoring systems and adaptive learning environments. Before the system can select any adaptation action like selecting tasks, learning material, or advice, it should first classify the learner's current situation. For this purpose, we need a *classifier* – a model, which predicts the class value from other *explanatory* attributes. For example, one can derive the student's motivation level from her/his actions in the tutoring system or predict the students who are likely to fail or drop out from their task scores. Such predictions are equally useful in the traditional teaching, but computerized learning systems often serve larger classes and collect more data for deriving classifiers.

Classifiers can be designed manually, based on expert's knowledge, but nowadays it is more common to *learn* them from real data. The basic idea is the following: First, we have to choose the classification method, like decision

trees, Bayesian networks, or neural networks. Second, we need a sample of data, where all class values are known. The data is divided into two parts, a *training set* and a *test set*. The training set is given to a learning algorithm, which derives a classifier. Then the classifier is tested with the test set, where all class values are hidden. If the classifier classifies most cases in the test set correctly, we can assume that it works accurately also on the future data. On the other hand, if the classifier makes too many errors (misclassifications) in the test data, we can assume that it was a wrong model. A better model can be searched after modifying the data, changing the settings of the learning algorithm, or by using another classification method.

Typically the learning task – like any data mining task – is an iterative process, where one has to try different data manipulations, classification approaches, and algorithm settings, before a good classifier is found. However, there exists a vast amount of both practical and theoretical knowledge which can guide the search process. In this chapter, we try to summarize and apply this knowledge on the educational context and give good recipes how to succeed in classification.

The rest of the chapter is organized as follows: In Section 2, we survey the previous research where classifiers for educational purposes have been learnt from data. In Section 3, we recall the main principles affecting the model accuracy and give several guidelines for accurate classification. In Section 4, we introduce the main approaches for classification and analyze their suitability to the educational domain. The final conclusions are drawn in Section 5.

2 Background

We begin with a literature survey on how data-driven classification has been applied in the educational context. We consider four types of classification problems which have often occurred in the previous research. For each group of experiments, we describe the classification problems solved, type and size of data, main classification methods, and achieved accuracy (expected proportion of correct classifications in the future data).

2.1 Predicting academic success

The first group consists of experiments where the task was to classify the student's academic success in the university level. The objectives were to predict drops-outs in the beginning of studies (Dekker et al., 2009; Superby et al., 2006; Herzog, 2006), graduation in time (Herzog, 2006; Barker et al., 2004), general performance (Nghe et al., 2007), or need for remedial classes (Ma et al., 2000).

The data sets were relatively large (500–20 000 rows, in average 7200 rows), because they were collected from the whole university or several universities, possibly during several years. The number of available attributes was also large (40–375), and only the most important were used. In addition to demographic data and course scores, the data often contained questionnaire data on student's perceptions, experiences, and financial situation.

All experiments compared several classification methods. Decision trees were the most common, but also Bayesian networks and neural networks were popular. The achieved accuracy was in average 79%, which is a good result for so difficult and important tasks. In the largest data sets (> 15 000 rows), 93–94% accuracy was achieved.

2.2 Predicting the course outcomes

The second group consists of experiments where the task was to classify the student's success in one course. The objectives were to predict passing/failing a course (Zang and Lin, 2003; Hämäläinen and Vinni, 2006; Hämäläinen et al., 2006; Bresfelean et al., 2008), drop-outs Kotsiantis et al. (2003), or the student's score (Minaei-Bidgoli et al., 2003; Romero et al., 2008; Mühlbrock, 2005). In most cases, the course was implemented as a distance learning course, where failure and drop-out are especially serious problems.

The data sets were relatively small (50–350 rows, in average 200), because they were restricted by the number of students who take the same course. Usually the data consisted of just one class of students, but if the course had remained unchanged, it was possible to pool data from several classes.

The main attributes concerned exercise tasks and the student's activity in the course, but also demographic and questionnaire data were used. The original number of attributes could be large (> 50), but was reduced to 3–10, before any model was learnt.

A large variety of classification methods were tried and compared in these experiments. The most common methods were decision trees, Bayesian networks, neural networks, K -nearest neighbour classifiers, and regression-based methods. The average accuracy was only 72%, but in the best cases nearly 90%. The most important factors affecting the classification accuracy were the number of class values used (best for the binary case) and at how early stage the predictions were done (best in the end of the course, when all attributes are available).

2.3 Succeeding in the next task

In the third group of experiments the task was to predict the student’s success in the next task, given her/his answers to previous tasks. This is an important problem especially in computerized adaptive testing, where the idea is to select the next question according to the student’s current knowledge level. Jonsson et al. (2005); Vomlel (2004); Desmarais and Pu (2005) predicted just the correctness of the student’s answer, while Liu (2000) predicted the student’s score in the next task.

The data sets were relatively small (40–360 rows, in average 130). The data consisted of students’ answers in the previous tasks (measured skill and achieved score) and possibly other attributes concerning the student’s activities in the learning system.

All experiments used probabilistic classification methods (Bayesian networks or Hidden Markov models). The accuracy was reported only in the last three experiments and varied between 73% and 90%.

2.4 Metacognitive skills, habits, and motivation

The fourth group covers experiments, where the task was to classify metacognitive skills and other factors which affect learning. The objectives were to predict the student’s motivation or engagement level (Cocea and Weibelzahl, 2006, 2007), cognitive style (Lee, 2001), expertise in using the learning system (Damez et al., 2005), “gaming” the system (Baker et al., 2004), or recommended intervention strategy (Hurley and Weibelzahl, 2007).

Real log data was used in the first five experiments. The size of data varied (30–950 rows, in average 160), because some experiments pooled all data on one student’s actions together, while others could use even short sequences of

sessions. The attributes concerned navigation habits, time spent in different activities, number of pages read, number of times a task was tried, etc. Only a small number of attributes (4–7) was used to learn models.

In Hurley and Weibelzahl (2007), a large set of artificial data was simulated. Four attributes were used to describe the student’s metacognitive skills (self efficacy, goal orientation, locus of control, perceived task difficulty). The idea was that later these attributes could be derived from log data.

The most common classification methods were decision trees, Bayesian networks, K -nearest neighbour classifiers, and regression-based techniques. Classification accuracy was reported only in four experiments and varied between 88% and 98%. One explanation for the high accuracy is that the class values were often decided by experts using some rules and the same attributes as the classifier used.

2.5 Summary

These 24 reviewed experiments give a good overview of the typical educational data and the most popular classification methods used.

In most cases the class attribute concerned a student, and there was just one row of data per each student. In the university level studies, the size of the data was still large, but in the course level studies, the data sets were small (50–350 rows). Larger data sets were available for tasks, where each sequence of log data was classified separately.

Typically, the original data contained both categorical and numeric attributes. Often, the data was discretized before modelling, but sometimes both numeric and categorical versions of the data were modelled and compared. Purely numeric data occurred when all attributes were task scores or statistics on log data (frequencies of actions, time spent in actions). How-

ever, the task scores had often just a few values, and the data was discrete. This is an important feature, because different classification methods suit for discrete and continuous data.

The most common classification methods were decisions trees (16 experiments), Bayesian networks (13), neural networks (6), K -nearest neighbour classifiers (6), support vector machines (3), and different kinds of regression-based techniques (10).

3 Main principles

In this section, we discuss the general principles which affect the selection of classification method and achieved classification accuracy. The main concerns are whether to choose a discriminative or probabilistic classifier, how to estimate the real accuracy, the tradeoff between overfitting and underfitting, and the impact of data preprocessing.

3.1 Discriminative or probabilistic classifier?

The basic form of classifiers are called *discriminative*, because they determine just one class value for each row of data. If M is a classifier (model), $C = \{c_1, \dots, c_l\}$ the set of class values, and t a row of data, then the predicted class is $M(t) = c_i$ for just one i .

An alternative is a *probabilistic* classifier, which defines the probability of classes for all classified rows. Now $M(t) = [P(C = c_1|t), \dots, P(C = c_l|t)]$, where $P(C = c_i|t)$ is the probability that t belongs to class c_i .

Probabilistic classification contains more information, which can be useful in some applications. One example is the task where one should predict the student's performance in a course, before the course has finished. The

data often contains many inconsistent rows, where all other attribute values are the same, but the class values are different. Therefore, the class values cannot be determined accurately, and it is more informative for the course instructors to know how likely the student will pass the course. It can also be pedagogically wiser to tell the student that she or he has 48% probability to pass the course than to inform that she or he is going to fail.

Another example occurs in intelligent tutoring systems (or computerized adaptive testing) where one should select the most suitable action (next task) based on the learner's current situation. Now each row of data t describes the learner profile. For each situation (class c_i) there can be several recommendable actions b_j with probabilities $P(B = b_j|C = c_i)$ which tell how useful action b_j is in class c_i . Now we can easily calculate the total probability $P(B = b_j|t)$ that action b_j is useful, given learner profile t .

3.2 Classification accuracy

The classification accuracy in set r is measured by *classification rate*, which defines the proportion of correctly classified rows in set r . If the predicted class by classifier M for row t is $M(t)$ and the actual class is $C(t)$, then the accuracy is

$$cr = \frac{\#\text{rows in } r \text{ where } M(t) = C(t)}{\#\text{rows in } r},$$

where #rows is an abbreviation for the number of rows.

The *classification error* in set r is simply the proportion of misclassified rows in r : $err = 1 - cr$.

If the class value is binary (e.g. the student passes or fails the course), the rates have special names:

	predicted class c_1	predicted class c_2
true class c_1	true positive rate $\frac{\#\text{rows where } M(t)=c_1=C(t)}{\#\text{rows where } M(t)=c_1}$	false negative rate $\frac{\#\text{number of rows where } M(t)=c_2 \neq C(t)=c_1}{\#\text{rows where } M(t)=c_2}$
true class c_2	false positive rate $\frac{\#\text{rows where } M(t)=c_1 \neq C(t)=c_2}{\#\text{rows where } M(t)=c_1}$	true negative rate $\frac{\#\text{rows where } M(t)=c_2=C(t)}{\#\text{rows where } M(t)=c_2}$

If the accuracy in one class is more critical (e.g. all possible failures or drop-outs should be identified), we can often bias the model to minimize false positive (false negative) rate, in the cost of large false negative (positive) rate (see e.g. (Witten and Frank, 2005)[ch. 5.7]).

When r is the training set, the error is called the *training error*. If r has the same distribution as the whole population (e.g. all future students in a similar course), the training error gives also a good estimate for the *generalization error*. Unfortunately, this is seldom the case in the educational domain. The training sets are so small that they cannot capture the real distribution and the resulting classifier is seriously biased. Therefore, we should somehow estimate the generalization error on unseen data.

A common solution is to reserve a part of the data as a test set. However, if the data set is already small, it is not advisable to reduce the training set any more. In this case, *m-fold cross-validation* is a better solution. The idea is that we partition the original data set of size n to m disjoint subsets of size n/m . Then we reserve one subset for validation and learn the model with other $m - 1$ subsets. The procedure is repeated m times with different validation sets and finally we calculate the mean of classification errors. An extreme case is *leave-one-out* cross-validation, where just one row is saved for validation and the model is learnt from the rest $n - 1$ rows.

3.3 Overfitting

Overfitting is an important problem related to accuracy. Overfitting means that the model has fitted to the training data too much so that it expresses even the rarest special cases and errors in data. The resulting model is so specialized that it cannot generalize to future data. For example, a data set which was collected for predicting the student's success in a programming course contained one female student, who had good it-skills and self-efficacy, and knew the idea of programming beforehand, but still dropped out the course. Still, we could not assume that all future students with the same characteristics would drop out. (In fact, all the other female students with good self-efficacy passed the course.)

Overfitting happens, when the model is too complex relative to the data size. The reason is that complex models have higher *representative power*, and they can represent all data peculiarities, including errors. On the other hand, simple models have lower representative power but they generalize well to future data. If the model is too simple, it cannot catch any essential patterns in the data, but *underfits*. It means that the model approximates poorly the true model or there does not exist any true model.

Figure 1 demonstrates the effects of overfitting and underfitting, when the model complexity increases. In this example, we used the previously mentioned data set from a programming course. The attributes were added to the model in the order of their importance (measured by Information Gain) and a decision tree was learnt with *ID3* algorithm. The simplest model used just one attribute (exercises points in applets), while the last model used 23 attributes. In the simplest models both the training and testing errors were large, because there were not enough attributes to discriminate the classes, and the models underfitted. On the other hand, the most complex

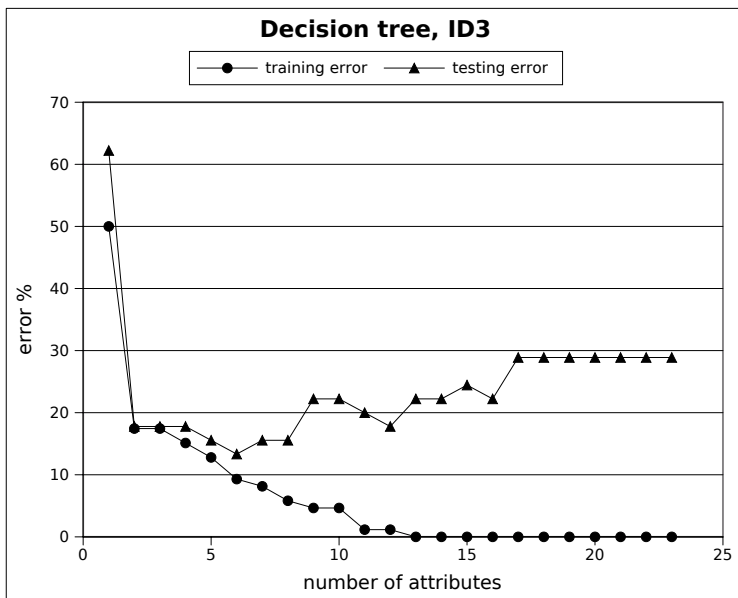


Figure 1: Effects of underfitting and overfitting. The training error decreases but testing error increases with the number of attributes.

models achieved a small training error, because they could fit the training data well. In the same time, the testing error increased, because the models had overfitted.

In the educational domain, overfitting is a critical problem, because there are many attributes available to construct a complex model, but only a little data to learn it accurately. As a rule of thumb it is often suggested (e.g. Jain et al. (2000); Duin (2000)) that we should have at least 5–10 rows data per each model parameter. The simpler the model is, the fewer parameters are needed. For example, if we have k binary-valued attributes, a naive Bayes classifier contains $O(k)$ parameters, while a general Bayesian classifier has in the worst case $O(2^k)$ parameters. In the first case, it is enough that $n > 5k$ ($10k$), while in the latter case, we need at least $n > 5 \cdot 2^k$ ($10 \cdot 2^k$) rows of data. If the attributes are not binary-valued, more data is needed.

In practice, there are two things we can do: 1) use simple classification methods requiring fewer model parameters, and 2) reduce the number of attributes and their domains by feature selection and feature extraction.

3.4 Linear and non-linear class boundaries

The main aspect of representative power is the form of class boundaries that can be represented. Linear classifiers can separate two classes only, if they are *linearly separable*, i.e. there exists a hyperplane (in two-dimensional case just a straight line) that separates the data points in both classes. Otherwise, the classes are *linearly inseparable* (Figure 2). It is still possible that only few data points are in the wrong side of the hyperplane, and thus the error in assuming a linear boundary is small. Depending on the degree of error, a linear classifier can still be preferable, because the resulting model is simpler and thus less sensitive to overfitting. However, some classes can be separated only by a non-linear boundary and a non-linear classifier is needed.

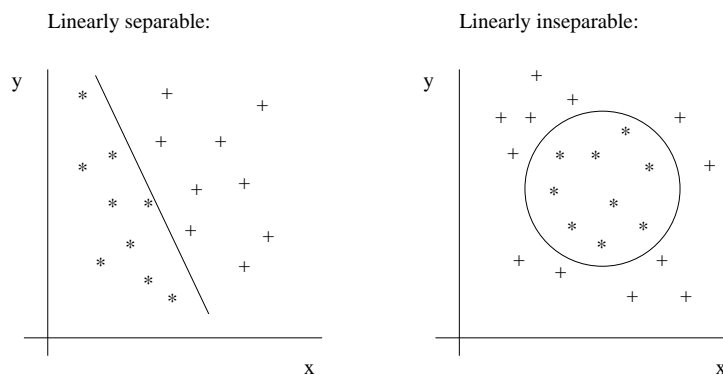


Figure 2: Linearly separable and inseparable class boundaries.

3.5 Data preprocessing

Before the modelling paradigm can be selected, we should know what kind of data is available. Typically, educational data is discrete-valued, either numeric or categorical. The numeric values like exercise points are either integers or represented with a low precision (e.g. 1/2 or 1/4 points). Log-based data can contain also continuous values like time spent on a task or average reading speed.

Real data is often incomplete in several ways. It can contain missing or erroneous attribute values. Erroneous values are generally called *noise*, but in some contexts noise means only measurement errors. Educational data is usually quite clean (free of noise), because it is either collected automatically (log data) or checked carefully (student's scores). Missing values occur more frequently, especially when we combine data from different sources. For example, questionnaire data on students' attitudes and habits could be combined with their performance data, but it is possible that some students have skipped the whole questionnaire or answered only some questions.

Outliers refer to data points which deviate significantly from the majority so that they do not fit the same model than others. Outliers can be due to noise, but in educational data they are often true observations. There are always exceptional students, who succeed with little effort or fail against all expectations.

The goal of data preprocessing is to improve the quality of data and produce good attributes for classification. The main tasks are *data cleaning*, *feature extraction*, and *feature selection*. In data cleaning, one should fill in the missing values and try to identify and correct errors. In feature extraction, new attributes are produced by transforming and combining the original ones. In feature selection, an optimal set of attributes is selected.

The most commonly used feature extraction technique in educational classification is discretization. In discretization, the range of numeric values is divided into intervals, which will be used as new attribute values. In the extreme case, all attributes can be binarized. For example, the exercise points can be divided into two categories, low (0) and high (1). Even if some information is lost, the resulting model can produce a more accurate classification. The reason is that small variation in exercise points or exact point values is not important, but the general tendency; if the student has done a little or a lot of them. Generally, discretization smooths out the effect of noise and enables simpler models, which are less prone to overfitting.

Feature selection can be done by analyzing the dependencies between the class attribute and explanatory attributes (e.g. correlation analysis, information gain, or association rules). Another approach is to use a learning algorithm and select the most important attributes in the resulting classifier. For example, one can first learn a decision tree and then use its attributes for a K -nearest neighbour classifier (Witten and Frank, 2005).

Some techniques perform feature extraction and feature selection simultaneously. For example, in *principal component analysis (PCA)* (Jolliffe, 1986) and *independent component analysis (ICA)* (Jutten and Herault, 1991) new attributes are produced as linear combinations of the original ones. In the same time, they suggest which of the new attributes describe the data best.

The dilemma is that the goodness of feature extraction and selection cannot be evaluated, before the classifier is learnt and tested. If the number of attributes is large, all possibilities cannot be tested. That is why all feature extraction and selection methods are more or less heuristic. Overviews of feature extraction and selection techniques can be found in (Han and Kamber, 2006)[ch 3] and (Witten and Frank, 2005)[ch 7.1–7.3].

4 Classification approaches

In the following, we will briefly introduce the main approaches for classification: decision trees, Bayesian classifiers, neural networks, nearest neighbour classifiers, support vector machines, and linear regression. The approaches are compared for their suitability to classify typical educational data.

4.1 Decision trees

Decision trees (see e.g. Witten and Frank (2005)[ch 6.1]) are maybe the best-known classification paradigm. A decision tree represents a set of classification rules in a tree form. Each root-leaf path corresponds to a rule of form $T_{i_1} \wedge \dots \wedge T_{i_l} \rightarrow (C = c)$, where c is the class value in the leaf and each T_{i_j} is a Boolean-valued test on attribute A_{i_j} .

The earliest decision trees were constructed by human experts, but nowadays they are usually learnt from data. The best known algorithms are *ID3* (Quinlan, 1986) and *C4.5* (Quinlan, 1993). The basic idea in all learning algorithms is to partition the attribute space until some termination criterion is reached in each leaf. Usually, the criterion is that all points in the leaf belong to one class. However, if the data contains inconsistencies, this is not possible. As a solution, the most common class among the data points in the leaf is selected. An alternative is to report the class probabilities according to relative frequencies in the node.

Decision trees have many advantages: they are simple and easy to understand, they can handle mixed variables (i.e. both numeric and categorical variables), they can classify new examples quickly, and they are flexible. Enlargements of decision trees can easily handle small noise and missing attribute values. Decision trees have high representative power, because they

can approximate non-linear class boundaries, even if the boundaries are everywhere piecewise parallel to attribute axes. However, it should be remembered that the resulting model can be seriously overfitted, especially if we have a small training set.

The main restriction of decision trees is the assumption that all data points in the domain can be classified deterministically into exactly one class. As a result all inconsistencies are interpreted as errors, and decision trees are not suitable for intrinsically non-deterministic domains. One such example is course performance data, where a significant proportion of rows can be inconsistent. Class probabilities have sometimes been suggested as a solution, but the resulting system is very unstable, because each leaf node has its own probability distribution (Hand et al., 2002)[p. 346]. Thus, even a minor change in one of the input variables can change the probabilities totally, when the data point is assigned to another leaf node.

Another problem is that decision trees are very sensitive to overfitting, especially in small data sets. In educational applications the future data seldom follows the same distribution as the training set and we would need more robust models. For example, Domingos and Pazzani (1997) recommend to use naive Bayes instead of decision trees for small data sets, even if the attributes were not independent, as naive Bayes assumes.

Often overfitting can be avoided if we learn a collection of decision trees and average their predictions. This approach is generally called *model averaging* or *ensemble learning* (see e.g. Valentini and Masulli (2002)). In ensemble learning we can combine several models with different structures, and even from different modelling paradigms. In practice, these methods can improve classification accuracy remarkably.

Finally, we recall that learning a globally optimal decision tree is an *NP*-

complete problem (Hyafil and Rivest, 1976). That is why all the common decision tree algorithms employ some heuristics and can produce suboptimal results.

4.2 Bayesian classifiers

In *Bayesian networks* (see e.g. Pearl (1988)) statistical dependencies are represented visually as a graph structure. The idea is that we take into account all information about conditional independencies and represent a minimal dependency structure of attributes. Each vertex in the graph corresponds to an attribute and the incoming edges define the set of attributes, on which it depends. The strength of dependencies is defined by conditional probabilities. For example, if A_1 depends on attributes A_2 and A_3 , the model has to define conditional probabilities $P(A_1|A_2, A_3)$ for all value combinations of A_1 , A_2 and A_3 .

When the Bayesian network is used for classification, we should first learn the dependency structure between explanatory attributes A_1, \dots, A_k and the class attribute C . In the educational technology, it has been quite common to define an ad hoc graph structure by experts. However, there is a high risk that the resulting network imposes irrelevant dependencies while skipping actually strong dependencies.

When the structure has been selected, the parameters are learnt from the data. The parameters define the class-conditional distributions $P(t|C = c)$ for all possible data points $t \in S$ and all class values c . When a new data point is classified, it is enough to calculate class probabilities $P(C = c|t)$ by the Bayes rule:

$$P(C = c|t) = \frac{P(C = c)P(t|C = c)}{P(t)}.$$

In practice, the problem is the large number of probabilities we have to estimate. For example, if all attributes A_1, \dots, A_k have v different values and all A_i s are mutually dependent, we have to define $O(v^k)$ probabilities. This means that we also need a large training set to estimate the required joint probability accurately.

Another problem which decreases the classification accuracy of Bayesian networks is the use of *Minimum Description Length (MDL)* score function for model selection (Friedman et al., 1997). *MDL* measures the error in the model over all variables, but it does not necessarily minimize the error in the class variable. This problem occurs especially, when the model contains several attributes and the accuracy of estimates $P(A_1, \dots, A_k)$ begins to dominate the score.

The *naive Bayes model* solves both problems. The model complexity is restricted by a strong independence assumption: we assume that all attributes A_1, \dots, A_k are conditionally independent, given the class attribute C , i.e. $P(A_1, \dots, A_k|C) = \prod_{i=1}^k P(A_i|C)$. This *Naive Bayes assumption* can be represented as a two-layer Bayesian network (Figure 3), with the class variable C as the root node and all the other variables A_1, \dots, A_k as leaf nodes. Now we have to estimate only $O(kv)$ probabilities per class. The use of *MDL* score function in the model selection is also avoided, because the model structure is fixed, once we have decided the explanatory variables A_i .

In practice, the Naive Bayes assumption holds very seldom, but still the naive Bayes classifiers have achieved good results. In fact, Domingos and Pazzani (1997) have shown that Naive Bayes assumption is only a sufficient but not a necessary condition for the optimality of the naive Bayes classifier. In addition, if we are only interested in the ranked order of the classes, it does not matter if the estimated probabilities are biased.

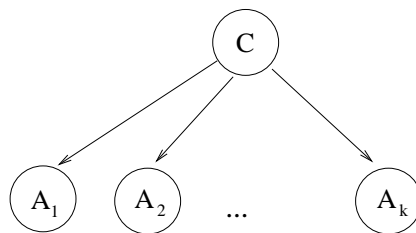


Figure 3: A naive Bayes model with class attribute C and explanatory attributes A_1, \dots, A_k .

As a consequence of Naive Bayes assumption, the representative power of the naive Bayes model is lower than that of decision trees. If the model uses nominal data, it can recognize only linear class boundaries. When numeric data is used, more complex (non-linear) boundaries can be represented.

Otherwise, the naive Bayes model has many advantages: it is very simple, efficient, robust to noise, and easy to interpret. It is especially suitable for small data sets, because it combines small complexity with a flexible probabilistic model. The basic model suits only for discrete data and the numeric data should be discretized. Alternatively, we can learn a continuous model by estimating densities instead of distributions. However, continuous Bayesian networks assume some general form of distribution, typically the normal distribution, which is often unrealistic. Usually, discretization is a better solution, because it also simplifies the model and the resulting classifier is more robust to overfitting.

4.3 Neural networks

Artificial neural networks (see e.g. Duda et al. (2000)) are very popular in pattern recognition, and justly so. According to a classical phrase (J. Denker, quoted in Russell and Norvig (2002)[585]), they are “the second best way of

doing just about anything”. Still, they can be problematic when applied to educational technology, unless you have a lot of numeric data and know exactly how to train the model.

Feed-forward neural networks (FFNN) are the most widely used type of neural networks. The *FFNN* architecture consists of layers of nodes: one for input nodes, one for output nodes, and at least one layer of *hidden nodes*. On each hidden layer the nodes are connected to the previous and next layer nodes and the edges are associated with individual weights. The most general model contains just one hidden layer. This is usually sufficient, because in principle *any function can be represented by a three-layer network*, given sufficiently many hidden nodes (Hecht-Nielsen, 1989). This implies that we can also represent any kind of (non-linear) class boundaries. However, in practice learning a highly non-linear network is very difficult or even impossible. For linearly separable classes it is sufficient to use a *perceptron*, a *FFNN* with no hidden layers.

The learning algorithm is an essential part of the neural network model. Even if neural networks can represent any kind of classifiers, we are seldom able to learn the optimal model. The learning is computationally hard and the results depend on several open parameters like the number of hidden layers, number of hidden nodes on each layer, initial weights, and the termination criterion. Especially the selection of the architecture (network topology) and the termination criterion are critical, because neural networks are very sensitive to overfitting. Unfortunately, there are no foolproof instructions and the parameters have to be defined by trial-and-error. However, there are some general rules of thumb, which restrict the number of trials needed. For example, Duda et al. (2000)[317] suggest to use a three-layer network as a default and add layers only for serious reasons. For stopping criterion

(deciding when the model is ready), a popular strategy is to use a separate test set (Mitchell, 1997)[111].

Feed-forward neural networks have several attractive features. They can easily learn non-linear boundaries and in principle represent any kind of classifiers. If the original variables are not discriminatory, *FFNN* transforms them implicitly. In addition, *FFNNs* are robust to noise and can be updated with new data.

The main disadvantage is that *FFNNs* need a lot of data – much more than typical educational data sets contain. They are very sensitive to overfitting and the problem is even more critical with small training sets. The data should be numeric and categorical data must be somehow quantized, before it can be used. However, this increases the model complexity and the results are sensitive to the quantization method used.

The neural network model is a black box and it is hard for people to understand the explanations for the outcomes. In addition, neural networks are unstable and achieve good results only in good hands (Duin, 2000). Finally, we recall that finding an optimal *FFNN* is an *NP*-complete problem (Blum and Rivest, 1988) and the learning algorithm can get stuck at a local optimum. Still, the training can be time consuming, especially if we want to circumvent overfitting.

4.4 *K*-nearest neighbour classifiers

K-nearest neighbour classifiers (see e.g. Hand et al. (2002)[347-352]) represent a totally different approach to classification. They do not build any explicit global model, but approximate it only locally and implicitly. The main idea is to classify a new object by examining the class values of the *K* most similar data points. The selected class can be either the most common

class among the neighbours or a class distribution in the neighbourhood.

The only learning task in K -nearest neighbour classifiers is to select two important parameters: the number of neighbours K and distance metric d .

An appropriate K value can be selected by trying different values and validating the results in a separate test set. When data sets are small, a good strategy is to use leave-one-out cross-validation. If K is fixed, then the size of the neighbourhood varies. In sparse areas the nearest neighbours are more remote than in dense areas. However, defining different K s for different areas is even more difficult. If K is very small, then the neighbourhood is also small and the classification is based on just a few data points. As a result the classifier is unstable, because these few neighbours can vary a lot. On the other hand, if K is very large, then the most likely class in the neighbourhood can deviate much from the real class. For small dimensional data sets suitable K is usually between 5 and 10. One solution is to weigh the neighbours by their distances. In this case, the neighbourhood can cover all data points so far and all neighbourhoods are equally large. The only disadvantage is that the computation becomes slower.

Defining the distance metric d is another, even a more critical problem. Usually, the metrics take into account all attributes, even if some attributes were irrelevant. Now it is possible that the most similar neighbours become remote and the "wrong neighbours" corrupt the classification. The problem becomes more serious, when more attributes are used and the attribute space is sparse. When all points are far, it is hard to recognize real neighbours from other points and the predictions become inaccurate. As a solution, it has been suggested (e.g. Hinneburg et al. (2000)) to give relevance weights for attributes, but the relevant attributes can also vary from class to class. In practice, appropriate feature selection can produce better results.

The nearest neighbourhood classifiers have several advantages: there are only two parameters to learn (or select), the classification accuracy can be very good in some problems, and the classification is quite robust to noise and missing values. Especially weighed distance smooths the noise in attribute values and missing values can be simply skipped. Nearest neighbour classifiers have very high representative power, because they can work with any kind of class boundaries, given sufficiently data.

The main disadvantage is the difficulty to select distance function d . Educational data often consists of both numeric and categorical data and numeric attributes can be in different scales. It means that we need a weighed distance function, but also a large data set to learn the weights accurately. Irrelevant attributes are also common in some educational data sets (e.g. questionnaire data) and they should be removed first.

The lack of an explicit model can be either an advantage or a disadvantage. If the model is very complex, it is often easier to approximate it only locally. In addition, there is no need to update the classifier, when new data is added. However, this kind of "lazy methods" are slower in classification than model-based approaches. If the data set is large, we need some index to find the nearest neighbours efficiently. It is also noteworthy that an explicit model is useful for human evaluators and designers of the system.

4.5 Support vector machines

Support vector machines (SVMs) (Vapnik, 1998) are an ideal method, when the class boundaries are non-linear but here is too little data to learn complex non-linear models. The underlying idea is that when the data is mapped to a higher dimension, the classes become linearly separable. In practice, the mapping is done only implicitly, using kernel functions.

SVMs concentrate on only the class boundaries; points which are any way easily classified, are skipped. The goal is to find the “thickest hyperplane” (with the largest margin), which separates the classes. Often, better results are achieved with “soft” margins, which allow some misclassified data points. When the optimal margin is determined, it is enough to save the *support vectors*, i.e. data points which define the class boundaries.

The main advantage of *SVMs* is that they find always the global optimum, because there are no local optima in maximizing the margin. Another benefit is that the accuracy does not depend on the dimensionality of data and the system is very robust to overfitting. This is an important advantage, when the class boundary is non-linear. Most other classification paradigms produce too complex models for non-linear boundaries.

However, *SVMs* have the same restriction as neural networks: the data should be continuous numerical (or quantized); the model is not easily interpreted, and selecting the appropriate parameters (especially the kernel function) can be difficult. Outliers can cause problems, because they are used to define the class borders. Usually, the problem is avoided by soft margins.

4.6 Linear regression

Linear regression is actually not a classification method, but it works well, when all attributes are numeric. For example, passing a course depends on the student’s points, and the points can be predicted by linear regression.

In linear regression, it is assumed that the target attribute (e.g. total points) is a linear function of other, mutually independent attributes. However, the model is very flexible and can work well, even if the actual dependency is only approximately linear or the other attributes are weakly

correlated (e.g. Xycoon (2000-2006)). The reason is that linear regression produces very simple models, which are not as risky for overfitting as more complex models. However, the data should not contain large gaps (empty areas) and the number of outliers should be small (Huber, 1981)[162].

4.7 Comparison

Selecting the most appropriate classification method for the given task is a difficult problem and no general answer can be given. In Table 1, we have evaluated the main classification methods according to eight general criteria, which are often relevant when educational data is classified.

The first criterion concerns the *form of class boundaries*. Decision trees, general Bayesian networks, *FFNNs*, nearest neighbour classifiers, and *SVMs* can represent highly non-linear boundaries. Naive Bayes model using nominal data can represent only a subset of linear boundaries, but with numeric data it can represent quite complex non-linear boundaries. Linear regression is restricted to only linear boundaries, but it tolerates small deviations from the linearity. It should be noticed that strong representative power is not desirable, if we have only little data and a simpler, linear model would suffice. The reason is that complex, non-linear models are also more sensitive to overfitting.

The second criterion, *accuracy on small data sets* is crucial for the educational domain. An accurate classifier cannot be learnt if there is not enough data. The sufficient amount of data depends on the model complexity. In practice, we should favour simple models, like naive Bayes classifiers or linear regression. Support vector machines can produce extremely good results, if the model parameters are just correctly selected. On the other hand, decision trees, *FFNNs*, and nearest neighbour classifiers require much larger data sets

Table 1: Comparison of different classification paradigms. Sign + means that the method supports the property, – that it does not. The abbreviations are *DT*=decision tree, *NB*=Naive Bayes classifier, *GB*=general Bayesian classifier, *FFNN*= feed-forward neural network, *K-nn*=*K*-nearest neighbour classifier, *SVM*=support vector machine, and *LR*= linear regression.

	<i>DT</i>	<i>NB</i>	<i>GB</i>	<i>FFNN</i>	<i>K-nn</i>	<i>SVM</i>	<i>LR</i>
Non-linear boundaries	+	(+)	+	+	+	+	–
Accuracy on small data sets	–	+	+/-	–	–	+	+
Works with incomplete data	–	+	+	+	+	–	–
Supports mixed variables	+	+	+	–	+	–	–
Natural interpretation	+	+	+	–	(+)	–	+
Efficient reasoning	+	+	+	+	–	+	+
Efficient learning	+/-	+	–	–	+/-	+	+
Efficient updating	–	+	+	+	+	–	+

to work accurately. The accuracy of general Bayesian classifiers depends on how complex structure is used.

The third criterion concerns whether the method can handle *incomplete data*, i.e. noise (errors), outliers (which can be due to noise), and missing values. Educational data is usually clean, but outliers and missing values occur frequently. Naive and general Bayesian classifiers, *FFNNs*, and nearest neighbour models are especially robust to noise in the data. Bayesian classifiers, nearest neighbour models, and some enlargements of decision trees can handle also missing values quite well. However, decision trees are generally very sensitive to small changes like noise in the data. Linear regression cannot handle missing attribute values at all and serious outliers can corrupt the whole model. *SVMs* are also sensitive to outliers.

The fourth criterion tells whether the method *supports mixed variables*, i.e. both numeric and categorical. All methods can handle numeric attributes, but categorical attributes are problematic for *FFNNs*, linear regression and *SVMs*.

Natural interpretation is also an important criterion, since all educational

models should be transparent to the learner (e.g. O’Shea et al. (1984)). All the other paradigms except neural networks and *SVMs* offer more or less understandable models. Especially decision trees and Bayesian networks have a comprehensive visual representation.

The last criteria concern the *computational efficiency* of classification, learning, and updating the model. The most important is efficient classification, because the system should adapt to the learner’s current situation immediately. For example, if the system offers individual exercises for learners, it should detect when easier or more challenging tasks are desired. Nearest neighbour classifier is the only one which lacks this property. The efficiency of learning is not so critical, because it is not done in real time. In some methods the models can be efficiently updated, given new data. This is an attractive feature because often we can collect new data when the model is already in use.

5 Conclusions

Classification has many applications in both traditional education and modern educational technology. The best results are achieved, when classifiers can be learnt from real data, but in educational domain the data sets are often too small for accurate learning.

In this chapter, we have discussed the main principles which affect classification accuracy. The most important concern is to select a sufficiently powerful model, which catches the dependencies between the class attribute and other attributes, but which is sufficiently simple to avoid overfitting. Both data preprocessing and the selected classification method affect this goal. To help the reader, we have analyzed the suitability of different classi-

fication methods for typical educational data and problems.

References

- Baker, R.S., A.T. Corbett, and K.R. Koedinger. 2004. Detecting student misuse of intelligent tutoring systems. In *Proceedings of the 7th international conference on intelligent tutoring systems (its'04)*, 531–540. Springer Verlag.
- Barker, K., T. Trafalis, and T.R. Rhoads. 2004. Learning from student data. In *Proceedings of the 2004 IEEE Systems and Information Engineering Design Symposium*, 79–86. Charlottesville, VA: University of Virginia.
- Blum, A., and R.L. Rivest. 1988. Training 3-node neural network is NP-complete. In *Proceedings of the 1988 Workshop on Computational Learning Theory (COLT)*, 9–18. MA, USA: MIT.
- Bresfelean, V.P., M. Bresfelean, N. Ghisoiu, and C.-A. Comes. 2008. Determining students academic failure profile founded on data mining methods. In *Proceedings of the 30th international conference on information technology interfaces (iti 2008)*, 317–322.
- Cocca, M., and S. Weibelzahl. 2006. Can log files analysis estimate learners' level of motivation? In *Proceedings of Lernen - Wissensentdeckung - Adaptivität (LWA2006)*, 32–35. Hildesheim.
- . 2007. Cross-system validation of engagement prediction from log files. In *Creating new learning experiences on a global scale, proceedings of the second european conference on technology enhanced learning (ectel2007)*, vol. 4753 of *Lecture Notes in Computer Science*, 14–25. Springer.

- Damez, M., T.H. Dang, C. Marsala, and B. Bouchon-Meunier. 2005. Fuzzy decision tree for user modeling from human-computer interactions. In *Proceedings of the 5th international conference on human system learning (ichsl'05)*, 287–302.
- Dekker, G., M. Pechenizkiy, and J. Vleeshouwers. 2009. Predicting students drop out: A case study. In *Educational data mining 2009: Proceedings of the 2nd international conference on educational data mining (edm'09)*, 41–50.
- Desmarais, M.C., and X. Pu. 2005. A Bayesian student model without hidden nodes and its comparison with item response theory. *International Journal of Artificial Intelligence in Education* 15:291–323.
- Domingos, P., and M. Pazzani. 1997. On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning* 29:103–130.
- Duda, R.O., P.E. Hart, and D.G. Stork. 2000. *Pattern classification*. 2nd ed. New Yor: Wiley-Interscience Publication.
- Duin, R. 2000. Learned from neural networks. In *Proceedings of the 6th annual conference of the advanced school for computing and imaging (asci-2000)*, 9–13. Advanced School for Computing and Imaging (ASCI).
- Friedman, N., D. Geiger, and M. Goldszmidt. 1997. Bayesian network classifiers. *Machine Learning* 29(2-3):131–163.
- Hämäläinen, W., T.H. Laine, and E. Sutinen. 2006. Data mining in personalizing distance education courses. In *Data mining in e-learning*, ed. C. Romero and S. Ventura, 157–171. Southampton, UK: WitPress.

- Hämäläinen, W., and M. Vinni. 2006. Comparison of machine learning methods for intelligent tutoring systems. In *Proceedings of the 8th international conference on intelligent tutoring systems*, vol. 4053 of *Lecture Notes in Computer Science*, 525–534. Springer-Verlag.
- Han, Jiawei, and Micheline Kamber. 2006. *Data mining: Concepts and techniques*. 2nd ed. Morgan Kaufmann.
- Hand, D., H. Mannila, and P. Smyth. 2002. *Principles of data mining*. Cambridge, Massachusetts, USA: MIT Press.
- Hecht-Nielsen, R. 1989. Theory of the backpropagation neural network. In *Proceedings of the international joint conference on neural networks (ijcnn)*, vol. 1, 593–605. IEEE.
- Herzog, S. 2006. Estimating student retention and degree-completion time: Decision trees and neural networks vis-a-vis regression. *New Directions for Institutional Research* 17–33.
- Hinneburg, A., C.C. Aggarwal, and D.A. Kleim. 2000. What is the nearest neighbor in high dimensional spaces? In *Proceedings of 26th international conference on very large data bases (vldb 2000)*, 506–515. Morgan Kaufmann.
- Huber, P.J. 1981. *Robust statistics*. Wiley Series in Probability and Mathematical Statistics, New York: John Wiley & Sons.
- Hurley, T., and S. Weibelzahl. 2007. Eliciting adaptation knowledge from on-line tutors to increase motivation. In *Proceedings of 11th international conference on user modeling (um2007)*, vol. 4511 of *Lecture Notes in Artificial Intelligence*, 370–374. Berlin: Springer Verlag.

- Hyafil, L., and R.L. Rivest. 1976. Constructing optimal binary decision trees is NP-complete. *Information Processing Letters* 5(1).
- Jain, A.K., P.W. Duin, and J. Mao. 2000. Statistical pattern recognition: a review. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22(1):4–37.
- Jolliffe, Ian T. 1986. *Principal component analysis*. Springer-Verlag.
- Jonsson, A., J. Johns, H. Mehranian, I. Arroyo, B. Woolf, A.G. Barto, D. Fisher, and S. Mahadevan. 2005. Evaluating the feasibility of learning student models from data. In *Papers from the 2005 aaai workshop on educational data mining*, 1–6. Menlo Park, CA: AAAI Press.
- Jutten, Christian, and Jeanny Herault. 1991. An adaptive algorithm based on neuromimetic architecture. *Signal Processing* 24:1–10.
- Kotsiantis, S.B., C.J. Pierrakeas, and P.E. Pintelas. 2003. Preventing student dropout in distance learning using machine learning techniques. In *Proceedings of 7th international conference on knowledge-based intelligent information and engineering systems (kes-2003)*, vol. 2774 of *Lecture Notes in Computer Science*, 267–274. Springer-Verlag.
- Lee, M.-G. 2001. Profiling students’ adaption styles in web-based learning. *Computers & Education* 36:121–132.
- Liu, C.-C. 2000. Knowledge discovery from web portfolios: tools for learning performance assessment. Ph.D. thesis, Department of Computer Science Information Engineering Yuan Ze University, Taiwan.
- Ma, Y., B. Liu, C.K. Wong, P.S. Yu, and S.M. Lee. 2000. Targeting the right students using data mining. In *Proceedings of the sixth acm sigkdd*

- international conference on knowledge discovery and data mining (kdd'00)*, 457–464. New York, NY, USA: ACM Press.
- Minaei-Bidgoli, B., D.A. Kashy, G. Kortemeyer, and W. Punch. 2003. Predicting student performance: an application of data mining methods with an educational web-based system. In *Proceedings of 33rd frontiers in education conference*, T2A–13–T2A18.
- Mitchell, T.M. 1997. *Machine learning*. New York, NY, USA: McGraw-Hill Companies.
- Mühlenbrock, M. 2005. Automatic action analysis in an interactive learning environment. In *Proceedings of the workshop on usage analysis in learning systems at aied-2005*, 73–80.
- Nghe, N. Thai, P. Janecek, and P. Haddawy. 2007. A comparative analysis of techniques for predicting academic performance. In *Proceedings of the 37th conference on asee/ieee frontiers in education*, T2G–7–T2G–12.
- O'Shea, T., R. Bornat, B. Boulay, and M. Eisenstad. 1984. Tools for creating intelligent computer tutors. In *Proceedings of the international nato symposium on artificial and human intelligence*, 181–199. New York, NY, USA: Elsevier North-Holland, Inc.
- Pearl, J. 1988. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. San Mateo, California: Morgan Kaufman Publishers.
- Quinlan, J.R. 1986. Induction of decision trees. *Machine Learning* 1(1): 81–106.
- . 1993. *C4.5: programs for machine learning*. Morgan Kaufmann.

- Romero, C., S. Ventura, P.G. P.G. Espejo, and C. Hervás. 2008. Data mining algorithms to classify students. In *Educational data mining 2008: Proceedings of the 1st international conference on educational data mining*, 8–17.
- Russell, S.J., and P. Norvig. 2002. *Artificial intelligence: A modern approach*. 2nd ed. Prentice Hall.
- Superby, J.F., J-P. Vandamme, and N. Meskens. 2006. Determination of factors influencing the achievement of the first-year university students using data mining methods. In *Proceedings of the workshop on educational data mining at its'06*, 37–44.
- Valentini, G., and F. Masulli. 2002. *Ensembles of learning machines*, vol. 2486 of *Lecture Notes in Computer Science*, 3–22. Springer-Verlag. Invited Review.
- Vapnik, V.N. 1998. *Statistical learning theory*. John Wiley & Sons.
- Vomlel, J. 2004. Bayesian networks in educational testing. *International Journal of Uncertainty, Fuzziness and Knowledge Based Systems* 12(Supplementary Issue 1):83–100.
- Witten, Ian H., and Eibe Frank. 2005. *Data mining: Practical machine learning tools and techniques*. 2nd ed. San Francisco: Morgan Kaufmann.
- Xycoon. 2000-2006. Linear regression techniques. In *Statistics - Econometrics - Forecasting (Online Econometrics Textbook)*, chap. II. Office for Research Development and Education. Available on <http://www.xycoon.com/>. Retrieved 1.1. 2006.
- Zang, W., and F. Lin. 2003. Investigation of web-based teaching and learning by boosting algorithms. In *Proceedings of IEEE International Conference*

on Information Technology: Research and Education (ITRE 2003), 445–449.