ELSEVIER

Artificial Intelligence in Medicine 8 (1996) 431–451

Artificial Intelligence in Medicine

Machine learning in prognosis of the femoral neck fracture recovery

Matjaž Kukar[a], Igor Kononenko[a], Tomaž Silvester[b,*]

[a] *University of Ljubljana, Faculty of Computer and Information Science, Tržaška 25, Slovenia*
[b] *Medical Faculty, Zaloška 2, SI-61001 Ljubljana, Slovenia*

Received 15 August 1995; accepted 1 April 1996

Abstract

We compare the performance of several machine learning algorithms in the problem of prognostics of the femoral neck fracture recovery: the K-nearest neighbours algorithm, the semi-naive Bayesian classifier, backpropagation with weight elimination learning of the multilayered neural networks, the LFC (lookahead feature construction) algorithm, and the Assistant-I and Assistant-R algorithms for top down induction of decision trees using information gain and RELIEFF as search heuristics, respectively. We compare the prognostic accuracy and the explanation ability of different classifiers. Among the different algorithms the semi-naive Bayesian classifier and Assistant-R seem to be the most appropriate. We analyze the combination of decisions of several classifiers for solving prediction problems and show that the combined classifier improves both performance and the explanation ability.

Keywords: Learning from examples; Estimating attributes; Explanation ability; Impurity function; Empirical comparison; Multiple knowledge

* Corresponding author. Tel./fax: +386 61 1768386; e-mail: (matjaz.kukar, igor.kononenko)@fer.uni-lj.si.

PII 0933-3657(96)00351-X

1. Introduction

The femur is the longest and the strongest bone in the human body. Its upper part, the femoral neck, is often subject to severe fractures, especially where older people are concerned. Intracapsular fractures of the femoral neck may considerably deteriorate the quality of their life. The specific structure and blood supply to the upper part of the femur contribute to complications, especially aseptic head necrosis and pseudoarthrosis (non-union), which frequently occur in the course of treatment. Such injuries should therefore be treated as urgent and the patient examined and treated as soon as possible.

The system for predicting possible complications should serve as a tool in the process of treatment and in the follow-up period. It should therefore be beneficial in two ways:

(1) to point out the problematic patients and advise them to attend follow-up examinations frequently (at least for 2 years after the operation) to catch possible complications in time;
(2) to help physicians choose the therapy which would minimize the risk of subsequent complications.

In recent years many machine learning algorithms have been developed that can be used as efficient tools for the analysis of databases and for extracting the classification knowledge that can be used to solve new problems in the given problem domain [6,21,22]. There have been many applications of machine learning to medical diagnostic problems [7,8,15,17,24,27,31]. As medical prognosis is a difficult task for physicians due to large time delays (several years) in recognizing the correct prognosis [36], machine learning may be more useful for solving prognostic rather than diagnostic problems.

Besides prediction accuracy, the explanation ability of the classifier is also very important. To support the prognostic process in everyday practice, physicians need a classifier that is able to explain its decisions. Such transparent decisions are much more acceptable for physicians. On the basis of the explanation he or she may accept or reject the proposed decision and eventually require additional tests to (dis)confirm the hypothesis.

In this paper we compare the performance and the explanation ability of several different classifiers in the femoral neck fracture recovery problem. The algorithms are: the K-nearest neighbours algorithm, the semi-naive Bayesian classifier, back-propagation with weight elimination learning of the multilayered feedforward neural networks, the LFC (lookahead feature construction) algorithm, and the Assistant-I and Assistant-R algorithms for top down induction of decision trees using information gain and RELIEFF as search heuristics, respectively. Section 2 briefly describes the different classifiers. Section 3 describes the data we used in our experiments. In Section 4 we compare the performance of the classifiers in terms of prediction accuracy and information score, and in Section 5 we compare the explanation ability of various classifiers. Section 6 gives the results for combinations of different classifiers. Section 7 gives suggestions for applications of machine learning in medical diagnostic and prognostic problems.

2. The compared algorithms

2.1. Basic terminology

Suppose we have a certain number of instances (patients) with the already known correct class (in our case one of the complications in treatment of the femoral neck fracture). The task of the machine learning system is to automatically generate a model (a description) of the given data.

Every instance is described with a certain number of attributes. Each attribute has its own set of possible attribute values, which can be either discrete (descriptive, for example) or continuous (numerical data). One of the discrete attributes has a special meaning — it represents the class of an instance (a concept to which an instance belongs).

A classifier is a system that categorizes a given instance into one of the possible classes. A classifier can be created either by using classification knowledge provided by a human expert or it can extract the necessary knowledge from a set of already classified instances (the so-called training set). On the basis of the training instances, and with use of a machine learning algorithm, the classifier produces a classification procedure that can later be used for classification of previously unseen instances.

The trivial default classifier would simply classify every instance into majority class (the class which most frequently occurs in the training set).

2.2. Assistant-R and Assistant-I

Assistant-R is a reimplementation of the Assistant learning system for top down induction of decision trees [3]. The basic algorithm goes back to CLS (Concept Learning System) developed by Hunt et al. [9] and reimplemented by several authors (see [26] for an overview). The main features of the original Assistant are the following.

- Binarization of attributes. The algorithm generates binary decision trees by binarizing each attribute at each decision step. At each step the binarized version of each attribute is selected so that it maximizes the information gain of the attribute. For continuous attributes a decision point is selected that maximizes the attribute's information gain. For discrete attributes a heuristic greedy algorithm is used to find the locally best split of attribute's values into two subsets.
- Decision tree pruning. Prepruning and postpruning techniques are used for pruning off unreliable parts of decision trees. For prepruning, three user-defined thresholds are provided: minimal number of training instances, minimal attribute information gain and maximal probability of the majority class in the current node. For postpruning, the method developed by Niblett and Bratko [23] is used. It uses Laplace's law of succession for estimating the expected classification error of the current node committed by pruning or not pruning its subtree.
- Incomplete data handling. During learning, training instances with a missing value of the selected attribute are weighted with probabilities of each attribute's

value conditioned with the class label. During classification, instances with missing values are weighted with unconditional probabilities of the attribute's values.

- Naive Bayesian classifier. For each internal node in a decision tree eventually a third branch appears labelled with the attribute's values for which no training instances are available. For such null leaves, the naive Bayesian formula is used to calculate the probability distribution in the leaf by using only the attributes that appear in the path from the root to the leaf:

$$P(C|A_{root}..A_{leaf}) = P(C)\prod_{A\in A} Q_A \text{ where } Q_A = \frac{P(A|C)}{P(A)} = \frac{P(C|A)}{P(C)} \quad (1)$$

Note that this calculation is done off-line, i.e. during the learning phase. For classification, the null leaves are already labelled with the calculated class probability distribution and are used for classification in the same manner as ordinary leaves.

The main difference between Assistant and its reimplementation Assistant-R is that RELIEFF is used for attribute selection. RELIEFF [16] is an extension of RELIEF [10,11]. The key idea of RELIEF is to estimate attributes according to how well their values distinguish among the instances that are near to each other.

For that purpose, given an instance, RELIEF searches for its two nearest neighbours: one from the same class (called 'nearest hit') and the other from a different class (called 'nearest miss'). The original algorithm of RELIEF randomly selects n training instances, where n is a user-defined parameter, as follows:

```
set all weights W[A]: = 0.0;
for i: = 1 to n do
begin
  randomly select an instance R;
  find nearest hit H and nearest miss M;
  for A: = 1 to all_attributes do
  W[A]: = W[A] – diff(A,R,H)/n + diff(A,R,M)/n;
end;
```

Here 'diff(Attribute,Instance1,Instance2)' calculates the difference between the values of Attribute for two instances. For discrete attributes the difference is either 1 (the values are different) or 0 (the values are equal), while for continuous attributes the difference is the actual difference normalized to the interval [0,1]. Normalization with n guarantees that all weights are in the interval $[-1,1]$. The function 'diff' is used also for calculating the distance between instances to find the nearest neighbours. The total distance is simply the sum of differences over all attributes (in fact the original RELIEF uses the squared difference, which for discrete attributes is equivalent to 'diff'. In all our experiments, there was no significant difference between results using diff or squared difference).

The original RELIEF can deal with discrete and continuous attributes. However, it cannot deal with incomplete data and is limited to two-class problems only. Kononenko [16] showed that RELIEF's estimates are strongly related to impurity functions. He also developed an extension of RELIEF, called RELIEFF, that improves the original algorithm by estimating probabilities more reliably and extended it to deal with incomplete and multi-class data sets.

Another difference between Assistant and Assistant-R is that, wherever appropriate, instead of the relative frequency, Assistant-R uses the m-estimate of probabilities, which was shown to often significantly increase the performance of machine learning algorithms [1,2,5,34]. For prior probabilities the Laplace's law of succession is used:

$$P_a(X) = \frac{N(X) + 1}{N + \#\_of\_possible\_outcomes} \tag{2}$$

These prior probabilities are then used in the m-estimate of conditional probabilities:

$$P(X|Y) = \frac{N(X\&Y) + m \times P_a(X)}{N(Y) + m} = \frac{N(X\&Y)}{N(Y) + m} + \frac{m \times P_a(X)}{N(Y) + m} \tag{3}$$

The parameter m trades off between the contributions of the relative frequency and the prior probability. Both estimates are very useful especially when estimating probabilities of small datasets.

In our experiments, the parameter m was set to 2 (this setting is usually used as default and, empirically, gives satisfactory results [1,2] although with tuning better results may be expected). The m-estimate is used in the naive Bayesian formula (Eq. (1)), for postpruning instead of Laplace's law of succession as proposed by Cestnik and Bratko [2], and for RELIEFF's estimates of probabilities.

Assistant-I is a variant of Assistant-R that, instead of RELIEFF uses information gain for the selection criterion, as does Assistant. However, the other differences to Assistant remain (m-estimate of probabilities). This algorithm enables us to evaluate the contribution of RELIEFF. The parameters for Assistant-I and Assistant-R were fixed throughout the experiments (no prepruning, postpruning with $m = 2$).

2.3. K-nearest neighbours

For a given new instance this algorithm searches for the k nearest training instances and classifies the instance into the most frequent class of these k instances. For the K-nearest neighbours (K-NN) algorithm the same distance measure was used as for RELIEFF. The results presented were obtained using the Manhattan distance. Results using the Euclidian distance are practically the same. The best results with respect to the parameter k are presented, although for a fair comparison such parameter tuning should be allowed only on the training and not the testing sets.

2.4. The naive and the semi-naive Bayesian classifier

The naive Bayesian classifier uses the naive Bayes formula (Eq. (1)) to calculate the probability of each class given the values of all the attributes and assuming the conditional independence of the attributes. A new instance is classified into the class with maximum calculated probability. The m-estimate of probabilities was used (see Section 2.2) and the parameter m was set to 2 in all experiments.

The semi-naive Bayesian classifier [13] addresses the independence assumption, which is often not justified. Usually, the attributes are defined by a human (for example in medical data), and are therefore relatively independent, as humans tend to think linearly. However, it is not always so. To avoid this problem, the algorithm should detect the dependencies among attributes and join dependent attributes together. In addition, instead of joining whole attributes, only single values of different attributes can be joined, providing a more flexible solution.

The semi-naive Bayesian classifier, described in more detail in [13], tries to solve this trade-off between non-naivety and reliability of approximations of probabilities.

When calculating the probability of class C_j in Eq. (1) the influence of attributes A_i and A_l is defined by:

$$\frac{P(C_j|A_i)}{P(C_j)} \times \frac{P(C_j|A_l)}{P(C_j)} \tag{4}$$

If, instead of assuming the independence of values A_i and A_l, the values are inter-dependent, the corrected influence is given by:

$$\frac{P(C_j|A_iA_l)}{P(C_j)} \tag{5}$$

To join the two values two conditions should be satisfied: the values of Eqs. (4) and (5) should be sufficiently different while the approximation of $P(C_j|A_iA_l)$ remains sufficiently reliable.

2.5. Backpropagation with weight elimination

A multilayered feedforward neural network [32] is a hierarchical network consisting of fully interconnected layers of processing units (often called neurons). The output of each unit is connected to every unit in the next layer. A network consists of at least two layers — the input and the output. However, this kind of network is able to solve only a very limited class of problems. For a more general network the user has to specify the number and the size of layers between the input and output layers. Connections between units are often referred to as synapses, giving a loose analogy with brain structure. Each connection and unit has a real-valued weight or bias attached to it. The output of a neuron j for an instance p is:

$$out_{pj} = f(bias_j + \sum_i weight_{ji} \cdot output_{pi}) \tag{6}$$

with f being an activation function, usually the sigmoid function

$$f(x) = \frac{1}{1 + e^{-\theta x}} \tag{7}$$

The backpropagation learning procedure minimizes the squared error accumulated from all training instances $p \in P_L$:

$$E = \sum_{p \in P_L} \frac{1}{2} \sum_j (expected_{pj} - output_{pj})^2 \tag{8}$$

In each learning iteration the algorithm modifies weights according to the following rule:

$\Delta_p w_{ji} = \eta \delta_{pj} output_{pi}$, with δ_{pj} being

$$\delta_{pj} = \begin{cases} (expected_{pj} - output_{pj}) \cdot output_{pj}(1 - output_{pj}), & \text{for output units} \\ output_{pj}(1 - output_{pj}) \sum_k \delta_{pk} w_{kj}, & \text{for hidden units} \end{cases} \tag{9}$$

It can be shown [32] that modifying weights using the formula (Eq. (9)) implements a gradient descent on the error surface obtained from Eq. (8). Various improvements of the basic learning process are possible, such as including a 'momentum term' [32], which prevents the weight oscillations. The idea is to keep some inertia (proportional to the small real-valued parameter α) from the previous change of the weight:

$$\Delta w_{ij}^{(n+1)} = \eta \delta_j output_{pi} + \alpha \Delta w_{ij}^{(n)} \tag{10}$$

Perhaps the most annoying problem of backpropagation is 'overfitting' the training data. The trained network becomes too specialised for describing training instances and is unable to successfully classify unseen instances. This phenomenon is usually a consequence of using an oversized network with too many hidden units. Weight elimination [35] at least partially overcomes this problem. With a slight change in the error function (Eq. (8)), the network is forced to keep the weights as small as possible and possibly eliminate some of them.

$$E'_p = E_p + \lambda \sum_{i,j} \frac{\frac{w_{ij}^2}{w_0^2}}{1 + \frac{w_{ij}^2}{w_0^2}} \tag{11}$$

However, this change introduces a new parameter λ which needs to be large enough to eliminate redundant weights and small enough not to disturb the error surface too much.

2.6. LFC

Ragavan and Rendell [28], and Ragavan et al. [29] use limited lookahead in their LFC (Lookahead Feature Construction) algorithm for top down induction of decision trees to detect significant dependencies between attributes for constructive induction. They show interesting results on some data sets. The reimplementation that we used in our experiments was developed by Robnik [30].

LFC generates binary decision trees. At each node, the algorithm constructs new binary attributes from the original attributes, using logical operators (conjunction, disjunction, and negation). From the constructed binary attributes the best attribute is selected and the process is recursively repeated on two subsets of training instances, corresponding to the two values of the selected attribute.

A limited lookahead is used for constructive induction. The space of possible useful constructs is restricted, due to the geometrical representation of the conditional entropy which is the estimator of an attribute's quality. To further reduce the search space, the algorithm also limits the breadth and the depth of the search. To make results comparable to Assistant-R we equipped LFC with pruning and the probability estimation facilities as described in Section 2.2.

3. Description of the data

The data, originating from the University Trauma Clinic in Ljubljana, Slovenia, was provided by T. Silvester [33]. He examined medical records, follow-up results, and radiographs of 197 patients (134 women and 63 men) with fractures of the femoral neck (see Fig. 1), treated at the University Trauma Clinic in Ljubljana in 1987.

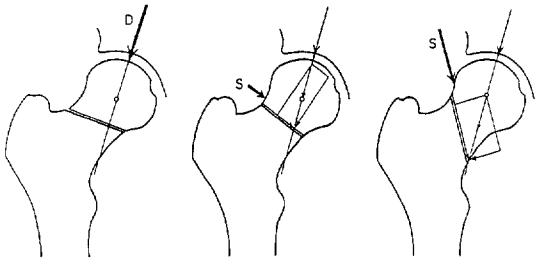

Fig. 1. A schematic representation of different femoral neck fractures.

Table 1
Possible complications for five- and two-class problems

Complications	Frequency	%
Redislocation	7	4.6
Protrusion of the blade plate in acetabulum	24	15.9
Asept. necrosis	13	8.6
Pseudoarthrosis	9	6.0
No complications	98	64.9
	Grouped ↓ complications	
Yes	53	35.1
No	98	64.9

Table 2
Domain description

Attribute	Numeric (range)	Discrete (no. of values)
Age	22 94	
Sex		2
Mobility before injury		7
State of health		7
Mechanism of injury		6
Additional injuries		9
Time between injury and examination (in h)	0.5–7200	
Time between injury and operation (in days)	0 35	
Fracture classification according to Garden		4
Fracture classification according to Pauwels		3
Therapy (type of operation)		10
Transfusion		2
Antibiotic prophylaxis		2
Anticoagulant therapy		2
Hospitalization time (in weeks)	0.5–40	
Hospital rehabilitation		2
General complications		10

Of the 197 patients, it was only known for 151 of them, what kind of complications, if any, had occurred. Originally, the patients were divided into five classes according to the complications which occurred during the treatment and follow-up period (Table 1). However, since some complications occur rarely, we also reduced the problem to predicting only whether complications will or will not occur. Even in this simplified form, it is still useful for physicians to discover problematic patients and observe them more carefully to prevent possible complications. Every patient was described with 17 continuous or unordered discrete attributes as listed in Table 2. On average, one attribute value per patient was unknown.

Table 3
Classification accuracy (%) in both prediction problems

Classifier	2 classes	5 classes
Backpropagation	71.1	65.3
Semi-naive Bayes	70.6	66.5
k-NN	69.0	65.0
Assistant-I	72.1	62.4
Assistant-R	70.8	63.6
LFC	69.6	65.3
Default	64.9	64.9
Physician (tree)	66.0	—

Table 4
Average information score (bit) in both prediction problems

Classifier	2 classes	5 classes
Backpropagation	0.35	0.40
Semi-naive Bayes	0.36	0.47
k-NN	0.40	0.46
Assistant-I	0.23	0.32
Assistant-R	0.25	0.33
LFC	0.41	0.45
Default	0.00	0.00
Physician (tree)	0.20	—

4. Experiments with different algorithms

4.1. Experimental methodology

We compared the performance of the algorithms on two variants of the prediction problem: one with two classes (complications/no complications) and the other with five classes. Altogether there were 151 descriptions of patients with known femoral neck fracture recovery result. Each experiment on each data set was performed 10 times by randomly selecting 70% of the instances for learning and 30% for testing and the results were averaged. Each system used the same subsets of instances for learning and for testing in order to provide the same experimental conditions.

Besides the classification accuracy, we also measured the average information score [12]. This measure eliminates the influence of prior probabilities and deals appropriately with the probabilistic answers of a classifier. The average information score is defined as:

$$Inf = \frac{\sum_{i=1}^{\#\ testing\ instances} Inf_i}{\#\ testing\ instances} \tag{12}$$

where the information score of the classification of the i-th testing instance is defined by:

$$Inf_i = \begin{cases} -\log_2 P(Cl_i) + \log_2 P'(Cl_i), & P'(Cl_i) \geq P(Cl_i) \\ -(-\log_2(1 - P(Cl_i)) + \log_2(1 - P'(Cl_i))), & P'(Cl_i) < P(Cl_i) \end{cases} \tag{13}$$

Cl_i is the correct class of the i-th testing instance, $P(Cl)$ is the prior probability of class Cl and $P'(Cl)$ the probability provided by the classifier.

4.2. Experimental results

The results of the experiments on both data sets are provided in Table 3 (classification accuracy) and Table 4 (information score).

All of the algorithms compared achieved roughly the same classification accuracy. However, the information score criterion favours the LFC and the k-NN methods. These two methods seem to be more determined in their correct answers and less determined in the incorrect ones than the other methods, which achieve approximately the same classification accuracy.

However, if we consider the fact that the majority class is almost 65%, the improvements in classification accuracy over the trivial default classifier (see Section 2.1) are not very impressive. On the other hand, the default classifier's information score would be 0.00. Therefore, the classifiers' answers provide useful information.

In Tables 3 and 4 we also provide the performance of a decision tree, derived by a domain expert — a physician. The tree is depicted in Fig. 6 and was derived for 2-class problem only. The performance in the tables was obtained by classifying all patients with the physician's tree.

At this point the goals set in Section 1 should be reconsidered.

(1) The compared methods seem to be useful in predicting possible complications. The variety of their decisions (and explanations) can help the physician to predict the most likely complication.

(2) Any method can also be used for choosing the therapy which maximizes the possibility of successful recovery. With a known patient description, one can select every possible therapy and observe the probability of successful recovery for each of them. The method giving the highest probability may be the best one. However, in our database, every patient with an artificial implant (artroplastic) recovered successfully, so most methods favoured the artroplastic therapy. Unfortunately, it is highly unlikely that a physician, when dealing with otherwise healthy, younger patients with good regenerative abilities, would use an artificial implant (artroplastic), although this method proved to be almost 100% successful when applied to older patients. This result is also consistent with Silvester's [33] suggestion that the artroplastic therapy should be applied more frequently to older patients and patients with biomechanically unsuitable fractures (such as Pauwels III or Garden IV).

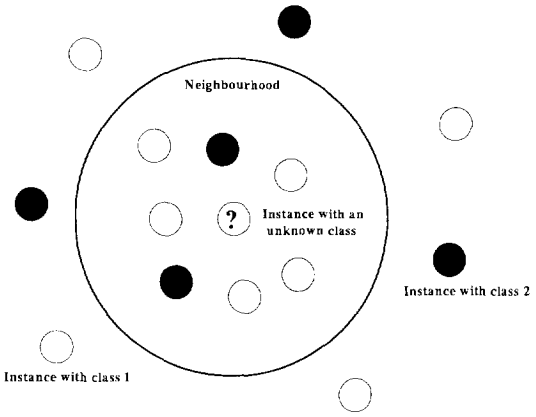

Fig. 2. K-nearest neighbours: explanation as a neighbourhood (K = 7).

Table 5
Semi-naive Bayes: an explanation of a decision

Decision = No complications (correct)		
Attribute value	For decision (bit)	Against decision (bit)
Age = 70–80	0.07	
Sex = Female		−0.19
Mobility before injury = Fully mobile	0.04	
State of health before injury = Other	0.52	
Mechanism of injury = Simple fall		−0.08
Additional injuries = None	0.00	
Time between injury and operation >10 days	0.42	
Fracture classification according to Garden = Garden III		−0.30
Fracture classification according to Pauwels = Pauwels III		−0.14
Transfusion = Yes	0.07	
Antibiotic prophylaxis = Yes		−0.32
Hospital rehabilitation = Yes	0.05	
General complications = None		−0.00
Combination:	0.21	
Time between injury and examination <6 h **and** Hospitalization time between 4 and 5 weeks		
Combination:	0.63	
Therapy = Artroplastic **and** Anticoagulant therapy = Yes		

5. Explanation abilities of different classifiers

5.1. General explanation abilities of different classifiers

In many fields of interest, and especially in medicine, it is very important for a domain expert to actually understand how and why the algorithm has made a certain decision. For the sake of comprehensibility lower classification accuracy may be acceptable. The explanation abilities of the algorithms are listed below.

- K-nearest neighbours: to explain the decision of the algorithm, a predefined number (K) of nearest neighbours from the training set is shown. This approach is very similar to the approach used by domain experts who make decisions on the basis of previously known similar cases (see Fig. 2).
- Naive and semi-naive Bayes: their decisions can be naturally interpreted as the sum of information gains [15]. For any attribute A from the set of all attributes A, the logarithm of $Q_A(C)$ from Eq. (1) can be calculated.

$$\log_2 Q_A(C) = -\log_2 P(C) - (-\log_2 P(C|A)) \tag{14}$$

This can be interpreted as the information gain contributed by the attribute A to the conclusion that an instance belongs to class C. If the posterior probability

$P(C|A)$ is greater than the prior probability $P(C)$, then the amount of information necessary to find that an instance belongs to class C, is given by:

$$-\log_2 P(C|A) = -\log_2 P(C) - \sum_{A \in A} \log_2 Q_A(C) \tag{15}$$

On the other hand, if $P(C|A) < P(C)$, the information is obtained in favour of class $\bar{C}$; i.e. the amount of information necessary to find that an instance does not belong to class C, is given with:

$$-\log_2 P(\bar{C}|A) = -\log_2 P(\bar{C}) - \sum_{A \in A} \log_2 Q_A(\bar{C}) \tag{16}$$

The expression -$\log_2 P(C)$ is the amount of information needed before classification and $\Sigma \log_2 Q_A$ is the sum of information gains from all attributes $A \in A$. In the case of the semi-naive Bayesian classifier, the process is exactly the same, except when the joint tuples of attribute/value pairs occur (see Eq. (5)). In this case, instead of simple attribute values, the joint values are used.

These information gains can be listed in a table to sum up the evidence for and against a decision. Table 5 provides a typical explanation of one decision. Each attribute has an associated strength, which is interpreted as the amount of information in bits contributed by that attribute. It can be in favour or against the classifier's decision.

One of the main advantages of such explanations is the usage of all the available attributes.

- Backpropagation neural networks and neural networks in general cannot easily explain their decisions. This is due to the large number of real-valued weights which all influence the result. In some cases it is possible to extract symbolic rules from the trained neural network. However, the rules tend to be large and relatively complex. Craven and Shavlik [4] compare rules extracted from a neural network with rules produced by Quinlan's C4.5 system [25]. The rules for a NetTalk data set extracted from a neural network have, on average, over 30

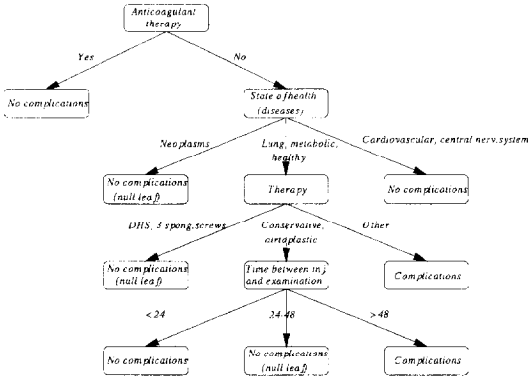

Fig. 3. Decision tree, generated by Assistant-I.

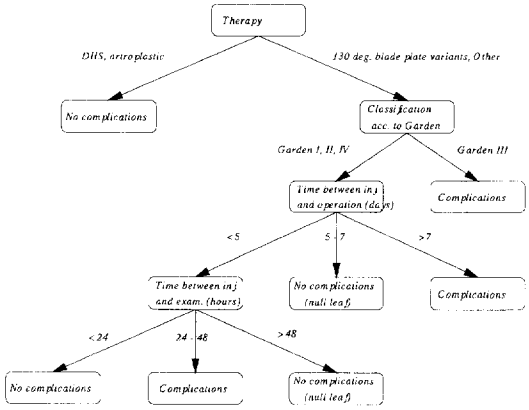

Fig. 4. Decision tree, generated by Assistant-R.

antecedents per rule compared to 2 antecedents for C4.5. Such rules are too complicated and hardly offer any explanation to a non-technically oriented domain expert.

- Decision trees (Assistant-I and Assistant-R): the explanation of a decision tree is the tree itself (see Figs. 3 and 4). It can be used without the computer and is fairly easy to understand. Positions of attributes in the tree, especially the top ones, often directly correspond to the domain expert's knowledge. However, in

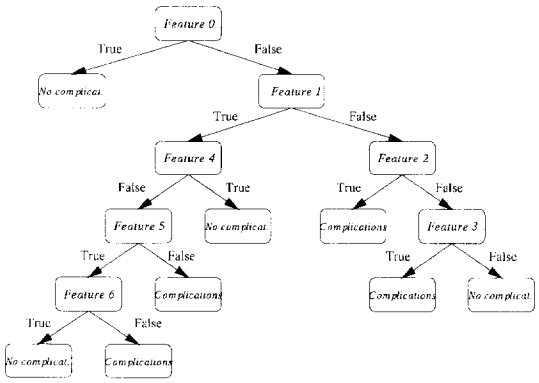

Fig. 5. Decision tree, generated by LFC. Feature 0: (Therapy = artroplastic) **or** (State of health = no serious illnesses) **or** (Mobility before injury ≠ fully mobile). Feature 1: (Classification according to Garden = Garden III) **and** (Anticoagulant therapy = no) **and** (State of health before injury ≠ healthy). Feature 2: (Classification according to Pauwels = Pauwels II) **or** (Classification according to Garden = Garden IV) **and** (Time between injury and operation > 10 days). Feature 3: (Time between injury and operation < 3 days) **and** (Time between injury to examination between 24 and 48 h). Feature 4: (Transfusion = yes) **or** (Hospital rehabilitation = no) **or** (Time between injury and operation between 5 and 7 days). Feature 5: (Time between injury and operation between 3 and 5 days) **and** (Therapy = 130 deg. blade plate with 1s + 1ss) **or** (Age > 80). Feature 6: (Time between injury and examination < 48 h) **and** (Mobility before injury = fully mobile).

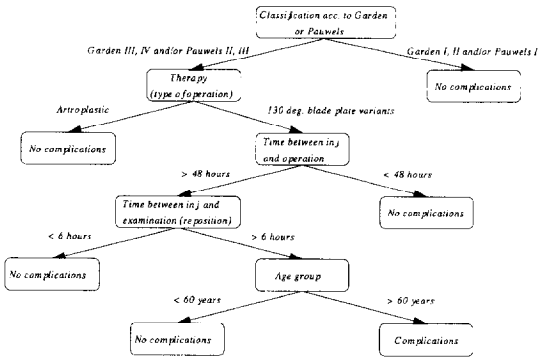

Fig. 6. Decision tree, as suggested by physicians for the two-class problem.

order to produce general rules, these methods use pruning which drastically reduces the tree size. Correspondingly, the paths from the root to the leaves are shorter, containing only few attributes, although they are the most informative. In many cases the domain experts feel that such a tree describes the patients too poorly to make reliable decisions [24].

- Lookahead feature construction (LFC) also generates decision trees. However, in each node a complex logical expression is used instead of a simple attribute value (Fig. 5). The generated trees can therefore be smaller and often easier to understand. The expressions often represent valid concepts from the domain theory. However, at the lower levels of the tree the expressions are often very specific and typically meaningless. Due to the complex logical expressions in nodes, the number of attributes used to classify an instance can be higher than in standard decision trees.

5.2. Explanation abilities evaluated by a domain expert

A domain expert, in our case a physician, evaluated all explanations offered by the algorithms compared. In the physician's opinion, the important attributes are the following:

- time passed between the moment of injury and medical examination,
- time passed between the moment of injury and operation,
- fracture classification according both to Garden and Pauwels,
- age and sex of the patient,
- kind of therapy applied to the patient.

Fig. 6 presents a decision tree, as suggested by the physicians.

Most of the above attributes appear in the algorithms' explanations. However, most of the algorithms do not include all of them. They even include attributes that physicians treat as unimportant for local status (e.g. anticoagulant therapy) and estimate them as more relevant than the ones listed above. Due to the fact that the physicians provided only a decision tree for the reduced problem (prognosing only whether the complications will or will not occur), all of the experiments described below were conducted on the two-class problem.

5.2.1. Semi-naive Bayesian classifier

The semi-naive Bayesian classifier has found 10 different combinations of attributes. However, the physician found only a few of them to be of interest. The interesting ones, which contributed strongly to the classification in the 'no complications' class were:

- Therapy = artroplastic **and** Anticoagulant therapy = yes. An artroplastic therapy has a high rate of success and it is usually accompanied by anticoagulant therapy.
- Classification according to Garden = Garden II **and** Classification according to Pauwels = Pauwels I. This one describes an 'easy' fracture which can be treated conservatively and often heals successfully.

5.2.2. Assistant-I and Assistant-R

Although Assistant-I achieved slightly better classification accuracy, the physicians found the decision tree generated by Assistant-R much more understandable since it contained 4 of the 5 attributes listed by the physicians (the tree generated by Assistant-I included only two of the above attributes). The physicians also suggested slightly different intervals for the attribute 'Time between injury and operation'. Instead of using intervals 'less than 5 days', '5–7 days' and 'more than 7 days', the suggested intervals were 'less than 3 days', '3–7 days' and 'more than 7 days'.

5.2.3. LFC

The lookahead feature construction algorithm found two features that directly represent the physicians' knowledge.

- (Classification according to Pauwels = Pauwels II **or** Classification according to Garden = Garden IV) **and** Time between injury and operation > 10 days. The first part of the conjunction describes a moderately to very complicated fracture.

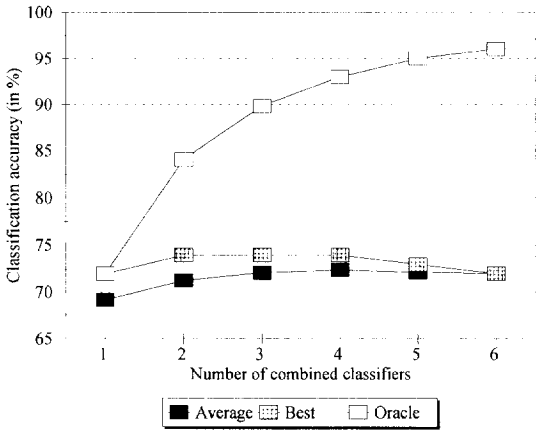

Fig. 7. Classification accuracy of a combined classifier.

The second part, time, means that the patient's state of health was not stable enough to perform the operation earlier or (s)he did not come to the doctor sooner (as is often the case with older, immobile patients). This long delay significantly contributes to possible complications.

- Time between injury and operation < 3 days **and** Time between injury and examination is 24–48 h. This feature describes a patient that was immediately examined and treated and therefore contributes to the classification in the 'No complications' class.

The physician's favourites seem to be the semi-naive Bayesian classifier and Assistant-R. LFC was also very interesting, however the physician claimed many features to be incomprehensible. The main advantage of the semi-naive Bayesian classifier is the inclusion of all attributes in the classification process. Decision trees are perhaps easier to understand, however the number and the choice of attributes are not always as the physicians would desire.

6. Combining decisions of different classifiers

In recent years the concepts of multistrategy learning and multiple knowledge have emerged. By this approach, the answers of several algorithms are combined, hoping that their combination will result in better performance. A combined system should also offer better explanation abilities and higher accuracy than any single classifier, no matter how finely tuned.

There are many different methods for combining decisions. However, in previous experiments it was shown [14,19] that in most cases the naive Bayesian combination outperforms other methods. Also, the explanation of the combined classifier is fairly straightforward. In essence, it is the same as that of the naive Bayesian classifier; except that instead of attributes we have the classifiers. Such 'meta level' explanations were also declared to be easily understandable by the physicians.

6.1. Naive Bayesian combination

Suppose we combine classifications of algorithms Alg_1, Alg_2,..., Alg_k. The classification of each algorithm is in the form of conditional probabilities $P(C_j|Alg_i)$ for every class C_j.

When the naive Bayesian combination [14,34] is used, the probability of an instance belonging to the class C_j is calculated with:

$$P(C_j|Alg_1,\ldots,Alg_k) = P(C_j)\prod_{i=1}^{k}\frac{P(C_j|Alg_i)}{P(C_j)} \tag{17}$$

6.2. Experimental results of combinations

To evaluate the combined classifier, we introduce an estimate for the upper bound of what a combined classifier could achieve. A combined classifier 'Oracle'

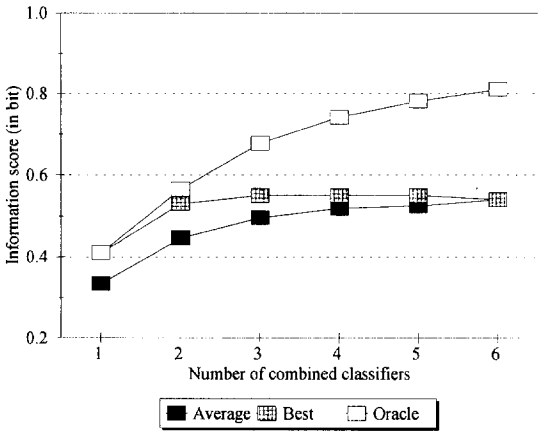

Fig. 8. Information score of a combined classifier.

combines classifications of single classifiers in the following way. For the final classification it chooses the classifier that maximizes the probability $P(C_j)$ for a known correct class C_j over classifications of every algorithm available in the current step. Note that a fair classifier should not have access to this information, however with the ideal combination function, a combined classifier could come close to the Oracle's performance.

The classification accuracy and the information score are shown in Figs. 7 and 8, respectively. The value on the X-axis of every figure is the number of the combined classifiers. At every point we combine any N classifiers from the 6 possible. Every figure contains three plots:

- 'Average' contains in each point the average of the classification accuracy or the information score of the $\binom{6}{N}$ possible combinations of N classifiers, combined with the naive Bayesian formula.
- 'Best' represents the best combined (using the naive Bayesian formula) classifier from the $\binom{6}{N}$ available. The best one is chosen on the basis of results on the test set and serves merely as an illustration of what the naive Bayesian combination could achieve.
- 'Oracle' represents the upper bound estimate. Note that by definition Best and Oracle coincide for $N = 1$, and Best and Average coincide for $N = 6$.

It is interesting to note that while the average classification accuracy increases only from 70.5% to 72.0%, the improvement of the information score is much more notable. On average it improves from 0.33 bits (an average of information scores of single classifiers) up to 0.54 bits (for 65%). Even when compared to the best single classifier, the improvement still remains impressive (from 0.41 bits up to 0.54 bits). The answers of combined classifiers, although not much more accurate, are therefore more informative and have more persuasive power. This is especially important when explaining classifications.

Another interesting thing to note is that by increasing N (up to 4 and more in this case) the differences between the best and average combined classifiers become quite minor. Therefore it suffices to choose a certain number of classifiers while it is less important which of them are selected.

7. Discussion

Our experiments show that various classifiers achieve similar classification accuracy in the problem of predicting femoral neck fracture recovery. Although the accuracy is not much higher than the default accuracy, the information score achieved shows that the information provided by the classifiers is useful. The explanation ability of different classifiers vary, being the highest for the semi-naive Bayesian classifier and the decision tree classifiers: Assistant-R, LFC, and Assistant-I. The results also indicate that the RELIEFF heuristic is better suited for generating human-understandable rules than the information gain heuristic. The explanation ability of the K-NN classifier is modest while that of the multilayered neural network is poor.

The general idea of finding a single best theory with machine learning algorithms goes back to the famous sentence due to William of Occam: 'It is vain to do with more what can be done with fewer' (Occam's razor). It is also formally founded by the minimal description length principle [20]. The simplest among theories with the identical performance is also the most probable and the most transparent due to its simplicity. However, the optimal prediction does not use just the single best theory but rather all possible theories. The final decision should be the weighted combinations of decisions of different theories where the weights correspond to probabilities of the theories given the evidence [20]. This 'principle of multiple explanations' is useful also in the cases where the explanation of a decision may be even more important than the accuracy of the decision.

Our experiments confirm the multiple explanation principle. Although the probabilities of different theories (classifiers) are not known, the naive Bayesian combination of decisions of various classifiers achieves more reliable classification in terms of the increased classification accuracy and the increased information score. In addition, the explanation ability of the combined classifier is much better. It can, at a meta-level, show the information provided by any single classifier, and can, at the single classifier level, provide the explanation of a decision of one particular classifier. Such different points of view for the same problem may provide further insight to the physicians when faced with a hard decision problem.

Acknowledgements

We thank Prof. Dr. Andrej Baraga and As. Dr. France Zupančič from the University Traumatology Clinic in Ljubljana for enabling access to the data and commenting on preliminary results in [18]. Assistant-I and Assistant-R were

implemented by Edvard Šimec, and LFC was reimplemented by Marko Robnik. This work was supported by the Slovenian Ministry of Science and Technology.

References

[1] B. Cestnik, Estimating probabilities: a crucial task in machine learning, in: *Proc. European Conference on Artificial Intelligence* (Stockholm, Sweden, 1990) 147–149.

[2] B. Cestnik and I. Bratko, On estimating probabilities in tree pruning, in: Y. Kodratoff, ed., *Proc. European Working Session on Learning, Porto, Portugal* (Springer-Verlag, 1991) 138–150.

[3] B. Cestnik, I. Kononenko and I. Bratko, ASSISTANT 86: A knowledge elicitation tool for sophisticated users, in: I. Bratko and N. Lavrač, eds., *Progress in Machine Learning* (Sigma Press, Wilmslow, England, 1987).

[4] M.W. Craven and J.W. Shavlik, Learning symbolic rules using artificial neural networks, *Proc. 10th Int. Conf. on Machine Learning* (Amherst, MA, Morgan Kaufmann, 1993) 73–80.

[5] J. Cussens, Bayes and pseudo-Bayes estimates of conditional probabilities and their reliability, in: *Proc. European Conf. on Machine Learning, Vienna, Austria* (Springer-Verlag, 1993) 136–152.

[6] T.G. Dietterich and J.W. Shavlik, *Readings in machine learning* (Morgan Kaufmann, 1990).

[7] S. Hojker, I. Kononenko, A. Jauk, V. Fidler and M. Porenta, Expert system's development in the management of thyroid diseases, in: *Proc. European Congress for Nuclear Medicine* (Milano, Italy, 1988).

[8] K.A. Horn, P. Compton, L. Lazarus and J.R. Quinlan, An expert system for the interpretation of thyroid assays in a clinical laboratory, *Aust. Computer J.* 17 (1985) 7–11.

[9] E. Hunt, J. Martin and P. Stone, *Experiments in Induction* (Academic Press, New York, 1966).

[10] K. Kira and L. Rendell, A practical approach to feature selection, in: D. Sleeman and P. Edwards, eds., *Proc. Int. Conf. on Machine Learning, Aberdeen, UK* (Morgan Kaufmann, 1992) 249–256.

[11] K. Kira and L. Rendell, The feature selection problem: traditional methods and new algorithm, *Proc. AAAI '92* (San Jose, CA, 1992).

[12] I. Kononenko and I. Bratko, Information based evaluation criterion for classifier's performance, *Machine Learning* 6 (1991) 67–80.

[13] I. Kononenko, Semi-naive Bayesian classifier, in: Y. Kodratoff, ed., *Proc. European Working Session on Learning-91, Porto, Portugal*, (Springer-Verlag, 1991) 206–219.

[14] I. Kononenko, Combining decisions of multiple rules, in: B. du Boulayand and V. Sgorev, eds., *Artificial intelligence V: methodology, systems, applications* (Elsevier Science Publications, 1992).

[15] I. Kononenko, Inductive and Bayesian learning in medical diagnosis, *Appl. Artif. Intell.* 7 (1993) 317-337.

[16] I. Kononenko, Estimating attributes: analysis and extensions of RELIEF, in: L. De Raedt and F. Bergadano, eds., *Proc. European Conf. on Machine Learning, Catania, Italy*, (Springer-Verlag, 1994) 171–182.

[17] I. Kononenko, I. Bratko and E. Roškar, Experiments in automatic learning of medical diagnostic rules, in: *International School for the Synthesis of Expert's Knowledge Workshop* (Bled, Slovenia, 1984).

[18] M. Kukar, *An application of machine learning in the femoral neck fracture diagnosis*, B.Sc. Thesis, University of Ljubljana, Faculty of Electrical Eng. and Computer Science, Ljubljana, Slovenia, 1993. In Slovene.

[19] M. Kukar, Multistrategy attribute learning, in: *Proc. 3rd Electrotechnical and Computer Science Conference ERK'94* (Portorož, Slovenia, 1994). In Slovene.

[20] M. Li and P. Vitányi, *An introduction to Kolmogorov complexity and its applications* (Springer-Verlag, New York, 1993).

[21] R.S. Michalski, J.G. Carbonell and T.M. Mitchell, *Machine Learning: An Artificial Intelligence Approach* (Tioga Publ. Co., 1983).

[22] R.S. Michalski, J.G. Carbonell and T.M. Mitchell, *Machine Learning: An Artificial Intelligence Approach*, Vol. II, (Morgan Kaufmann, 1986).
[23] T. Niblett and I. Bratko, Learning decision rules in noisy domains, in: *Proc. Expert Systems 86*, Brighton, UK, 1986.
[24] V. Pirnat, I. Kononenko, T. Janc and I. Bratko, Medical estimation of automatically induced decision rules, in: *Proc. 2nd Europ. Conf. on Artificial Intelligence in Medicine* (City University, London, 1989) 24–36.
[25] J.R. Quinlan, *C4.5: Programs for Machine Learning* (Morgan Kaufmann, San Mateo, CA, 1993).
[26] J.R. Quinlan, Induction of decision trees, *Machine Learning* 1 (1986) 81–106.
[27] J.R. Quinlan, P. Compton, K.A. Horn and L. Lazarus, Inductive knowledge acquisition: a case study, in: J.R. Quinlan, ed., *Applications of expert systems* (Turing Institute Press & Addison-Wesley; also in *Proc. 2nd Australian Conf. on Applications of Expert Systems*, Sydney, May 1986).
[28] H. Ragavan and L. Rendell, Lookahead feature construction for learning hard concepts, *Proc. 10th Int. Conf. Machine Learning, Amherst, MA* (Morgan Kaufmann, 1993) 252–259.
[29] H. Ragavan, L. Rendell, M. Shaw and A. Tessmer, Learning complex real-world concepts through feature construction, Technical Report UIUC-BI-AI-93-03, The Beckman Institute, University of Illinois, 1993.
[30] M. Robnik, *Constructive induction with decision trees*, B.Sc. Thesis, University of Ljubljana, Faculty of Electrical Eng. and Computer Science, Ljubljana, Slovenia, 1993. In Slovene.
[31] E. Roškar, P. Abrams, I. Bratko and I. Kononenko, MCUDS — an expert system for the diagnostics of lower urinary tract disorders, *J. Biomed. Meas. Informatics Control* 1 (1986) 201–204.
[32] D.E. Rumelhart and J.L. McClelland, *Parallel Distributed Processing, Vol. 1: Foundations* (MIT Press, Cambridge, 1986).
[33] T. Silvester, The rate of femoral head necrosis depending on the time interval from the injury to the internal fixation, *Medicinski razgledi 1992* 31 (1992) 293–313. In Slovene.
[34] P. Smyth, R.M. Goodman and C. Higgins, A hybrid rule-based bayesian classifier, in: *Proc. European Conf. on Artificial Intelligence* (Stockholm, Sweden, 1990) 610–615.
[35] S. Weigand, A. Huberman and D.E. Rumelhart, Predicting the future: a connectionist approach, *Int. J. Neural Syst.* 1 (1990) 193–209.
[36] M. Zwitter, I. Bratko and I. Kononenko, Rational and irrational reservations against the use of computer in medical diagnosis and prognosis, in: *Proc. 3rd Mediterranean Conf. on Medical and Biological Engineering*, Portorož, Slovenia, 1983.