Improving Stability of Decision Trees

Mark Last[1] mlast@bgumail.bgu.ac.il
Department of Information Systems Engineering, Ben-Gurion University of the Negev, Beer-Sheva 84105, Israel

Oded Maimon maimon@eng.tau.ac.il
Department of Industrial Engineering, Tel-Aviv University, Israel

Einat Minkov einat.minkov@amdocs.com
Amdocs Israel

Abstract

Decision-tree algorithms are known to be unstable: small variations in the training set can result in different trees and different predictions for the same validation examples. Both accuracy and stability can be improved by learning *multiple models* from bootstrap samples of training data, but the "meta-learner" approach makes the extracted knowledge hardly interpretable. In the following paper, we present the Info-Fuzzy Network (IFN), a novel information-theoretic method for building stable and comprehensible decision-tree models. The stability of the IFN algorithm is ensured by restricting the tree structure to using the same feature for all nodes of the same tree level and by the built-in statistical significance tests. The IFN method is shown empirically to produce more compact and stable models than the "meta-learner" techniques, while preserving a reasonable level of predictive accuracy.

Keywords. Decision trees, info-fuzzy network, stability, similarity, classification accuracy, output complexity, multiple models.

[1] Corresponding author

1. Introduction

As indicated by Breiman et al. (1984), decision-tree models have two goals: producing an accurate classifier and understanding the predictive structure of the problem. Though the classification accuracy of decision trees has been a subject of numerous studies, the second goal is also important for the knowledge discovery process, which focuses on "finding understandable patterns that can be interpreted as useful or interesting knowledge" (Fayyad et al., 1996). Consequently, several methods (like C4.5RULES by Quinlan, 1993) have been developed for converting a decision tree into a set of interpretable rules.

The patterns induced from the training data by a data mining algorithm are expected to be *valid* on a new sample extracted from the same population with some degree of certainty (Fayyad et al., 1996). The assumption of algorithm stability in terms of overall misclassification rates is the basis for the very common method of cross-validation, where multiple classifiers are constructed by using partially overlapping subsets of the training set (see Breiman et al., 1984). However, as pointed out by the authors of the leading books on decision tree learning (Breiman et al., 1984 and Quinlan, 1993), existing methods of constructing decision trees from data suffer from a major problem of instability. The symptoms of instability include variations in the predictive accuracy of the model and in the model topology. Instability can be revealed not only by using disjoint sets of data, but even by replacing a small portion of training cases, like in the cross-validation procedure. If an algorithm is unstable, the cross-validation results become estimators with high variance (Liu and Motoda, 1998), which means that an algorithm fails to make a clear distinction between persistent and random patterns in the data, a phenomenon known as *overfitting* (see Mitchell, 1997).

Formally, *semantic stability* of a classification algorithm is defined by Turney (1995) as the degree to which an algorithm generates repeatable results, given different batches of data from the same process. In mathematical terms, stability is the expected agreement between two models on a random sample of the original data, where agreement on a specific example means that both models assign it to the same class. The instability problem raises questions about validity of a particular tree, provided as an output of a decision-tree algorithm. The users view the learning algorithm as an oracle. Obviously, it is difficult to trust an oracle that says something radically different each time you make a slight change in the data.

There are different ways of dealing with the issue of instability. Thus, Breiman et al. (1984) suggest that the sequence of alternative trees be inspected by experts, who can use their domain knowledge to select the best tree. This option is readily available with the CART™ method, which builds an "efficiency frontier" of decision trees. The C4.5 algorithm (Quinlan 1993) produces only one tree in each run. Consequently, Quinlan recommends the "windowing" approach, which generates a single classifier from alternative trees based on several samples. Obviously, both approaches require an extra computational and human effort.

Decision-tree methods are not the only unstable classifiers. According to Breiman (1996), neural nets and regression trees are also unstable while k-nearest neighbors algorithms are stable. Several methods have been developed for combining multiple models to make more stable and accurate predictions. One of the methods, *bagging* (see Breiman, 1996 and Domingos, 1997b), builds each model by producing a new training set, where the original examples may appear more than once or not appear at all. Another approach, *boosting* (Freund & Shapire, 1996), generates multiple classifiers by maintaining a weight for each instance. As indicated by Domingos (1997a and 1998), the main drawback of the "multiple models" approach is making the extracted knowledge hardly comprehensible: combined models tend to be much more complex than each single model.

One attempt to recover the lost comprehensibility of the multiple models is done by (Domingos, 1997a). According to Domingos' approach (called *Combined Multiple Models*, or *CMM*), a classifier is applied to a set of randomly generated examples, whose classes are predicted by the combined models. The resulting model may be more complex than those produced by each single model, but, as shown by Domingos, it is much smaller than the meta-learned models. Still, a significant computational effort is required for constructing a combined model. Domingos evaluates the CMM method by using the semantic measure of stability (Turney 1995).

Though semantic measures of stability are important for comparing performance of different types of classifiers (e.g., decision trees vs. Naïve Bayes), the user of a given learning algorithm may be interested in the *syntactic stability* as well. As indicated above, algorithms are expected to produce similar sets of patterns from training samples based on the same distribution. Since a concept is a function from the attribute space to a set of classes, syntactic stability is a sufficient condition for semantic stability (an identical concept will provide the same prediction for the same instance). However, the opposite is not true. An algorithm may be semantically stable and still base its predictions upon different concepts.

Syntactic similarity of decision trees is related to a well-known graph-theoretic problem of *tree matching* (e.g., see Kilpeläinen, 1992 and Pelillo, 1999). Tree matching is concerned with finding the instances of a given pattern tree in a given target tree (Kilpeläinen, 1992). Subtrees detected by graph matching may be much smaller than the size of the target tree and even their labels may be completely different. Decision tree performance depends mainly on the total number of nodes, the labeling of each internal node in terms of tested attributes and their values, and the labeling of terminal nodes in terms of predicted classes. If labeling is ignored, two trees of identical structure may produce completely different predictions for the same instances. Moreover, having one decision tree as a subtree of another decision tree may have a negligible effect on the overall tree performance. Consequently, the problems of tree matching, tree inclusion, and sub-tree isomorphism are different from the problem of decision-tree similarity.

In this paper, we evaluate syntactic complexity and semantic stability of a new classification algorithm, termed IFN for Info-Fuzzy Network, initially introduced by us in (Maimon and Last, 2000). The IFN method produces decision-tree models, which are based on a minimal number of predicting features. The induction procedure combines statistical significance testing (essential for the model stability) and dimensionality reduction, which keeps the model as simple as possible. The second letter in the method name ("F") stands for a post-processing module, based on fuzzy logic, which calculates reliability of target values (see details in Maimon and Last, 2000). Post-processing operations are beyond the scope of this paper, since they have no impact on the structure of the network and its stability.

An outline of the IFN method is presented in the next section (2), where it is also compared to similar techniques for constructing trees and networks. Afterward, we use benchmark data from (Domingos, 1998) to compare the performance of the IFN algorithm to meta-learning methods, which are known for their stability and accuracy. The paper concludes with summarizing the obtained results and discussing some directions for future research.

2. Info-Fuzzy Network

2.1 Connectionist Network Structure

We model the association between the input (predicting) attributes and the target (dependent) attribute by the *Information-Theoretic Fuzzy Network* (IFN). The components of the network include the following:

1) *I* - a subset of *input* (predicting) attributes used by the model. The network construction algorithm selects input attributes from a set *C* of *candidate input* attributes (available features).

2) |*I*| - total number of *hidden layers* (levels) in the network. Each hidden layer is uniquely associated with a single input attribute by representing the interaction of that attribute and the input attributes of the previous layers. The first layer (Layer 0) includes only the root node and is not associated with any input attribute. Unlike the decision-tree topology implemented by Breiman et al. (1984) and

Quinlan (1986 and 1993), the hypothesis space of the IFN method is limited to testing the same feature at all nodes of a given layer. A similar idea was previously used in Kohavi and Li (1995) for constructing "oblivious decision graphs." The unique features of our method are discussed in sub-section 2.5 below.

3) L_l - a subset of nodes z in a hidden layer l. Each node represents a conjunction of values of the first l input attributes in the network.

4) K - a subset of distinct *target* nodes in the network. If the target (dependent) attribute is nominal, each target node is associated with a distinct category or a class. For continuous target attributes, the target nodes represent disjoint intervals of the attribute domain. A target layer is missing in the standard decision trees, but it is included in decision graphs (Kohavi and Li, 1995) and artificial neural networks (see Mitchell, 1997).

5) (z, j)- a connection between a terminal (unsplit) node z and a target node j. Each connection represents a *probabilistic rule* of the form "*if node is z then the value of the target attribute is j with probability $P(V_j/z)$,*" and it has an information-theoretic weight associated with it (see Maimon and Last, 2000). The output of standard decision-tree methods (like CART™ and C4.5) contains deterministic classification rules only (one rule per each terminal node).

Figure 1 below shows an Info-Fuzzy Network induced from the Credit ("Australian") Dataset. The connectionist nature of the IFN model (each terminal node is connected to every target node) resembles the structure of multi-layer neural networks, which also have connections between input, hidden, and output nodes. Consequently, we define our system as a *network* and not as a *tree*. However, info-fuzzy networks differ from neural networks in that weights are defined only for the connections to the target layer (see Figure 1), while internal connections are associated with values or intervals of input attributes and do not have any weights at all. A neural network has, in contrast, a weight associated with every inter-layer connection.

A standard decision tree can easily be extracted from the IFN structure by removing the target layer and associating a single classification rule with each terminal node in the network (see sub-section 2.4 below). The induction algorithm for constructing an Info-Fuzzy Network from data is described in the next sub-section.

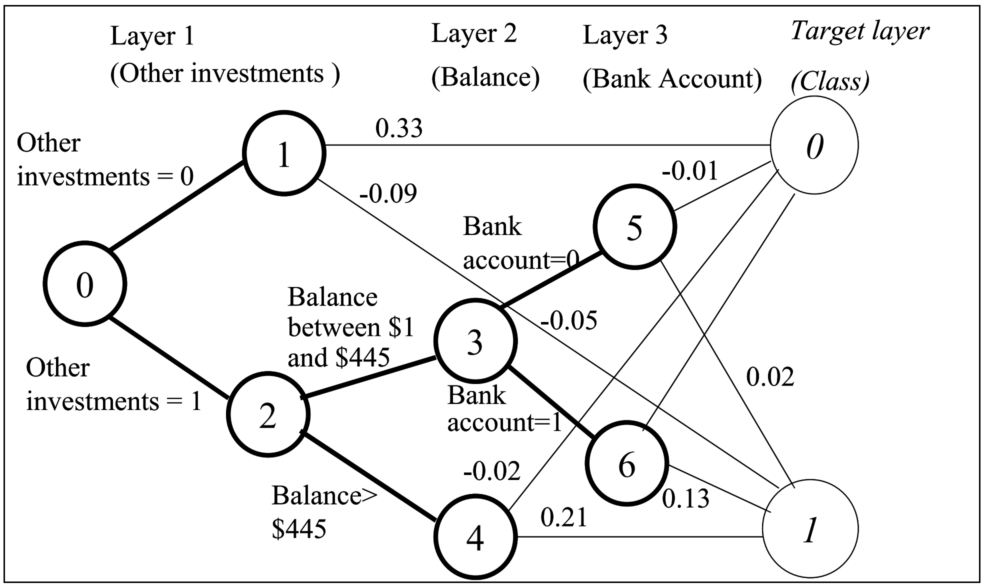

Figure 1 IFN Example 1: Credit Dataset

2.2 Network Construction Procedure

The network construction algorithm starts with defining the target layer, where each node stands for a distinct target value (class), and the “root” node representing an empty set of input attributes. The connections between the root node and the target nodes represent unconditional (prior) probabilities of the target values. Unlike CART (see Breiman et al. 1984) and C4.5 (Quinlan, 1993), IFN is built only in one direction (top-down). After the construction process is stopped, there is no bottom-up post-pruning of the network branches. The process of *pre-pruning* the network is explained below

A node is split only if it provides a statistically significant increase in the *mutual information* of the node and the target attribute. Mutual information, or information gain, is defined as a decrease in the conditional entropy of the target attribute (see Cover, 1991). If a tested feature is nominal, the splits correspond to the feature values. Splits on continuous features represent thresholds, which maximize an increase in mutual information. For each layer, the algorithm re-computes the best threshold splits of continuous attributes and chooses an input attribute (either discrete, or continuous), which provides the maximum overall increase in mutual information across all nodes of the final layer.

If the maximum increase in mutual information is greater than zero, a new hidden layer is added to the network. The nodes of a new layer are defined for a Cartesian product of split nodes of the previous final layer and the values of a new input attribute. According to the chain rule (see Cover, 1991), the mutual information between a set of input attributes and the target (defined as the overall decrease in the conditional entropy) is equal to the sum of drops in conditional entropy across all the hidden layers. If there is no candidate input attribute significantly decreasing the conditional entropy of the target attribute, no more layers are added and the network construction stops.

Information-theoretic criteria for choosing the best attribute are used by several decision-tree algorithms, like ID3 (Quinlan, 1986), C4.5 (Quinlan, 1993), CART (Breiman et al., 1984), and EODG (Kohavi and Li, 1995). However, the IFN algorithm, unlike most other methods, tests the *statistical significance* of the entropy change. We should note here that the significance of the chi-square test has been used as a stopping criterion in the original version of the ID3 algorithm (Quinlan, 1986), but later versions of Quinlan's method have abandoned this test by turning to the *post-pruning* approach (see Quinlan, 1993).

The main steps of the construction procedure are summarized in Table 1. Complete details are provided in (Maimon and Last, 2000).

Table 1 Network Construction Algorithm

Input:	The set of *n* training instances; the set *C* of candidate input attributes (discrete and continuous); the target (classification) attribute A_i; the minimum significance level *sign* for splitting a network node (default: *sign* = 0.1%).
Output:	A set *I* of selected input attributes and an information-theoretic network. Each selected input attribute has a corresponding hidden layer in the network.
Step 1	Initialize the information-theoretic network (single root node representing all records, no hidden layers, and a target layer for the values of the target attribute).
Step 2	While the number of layers $\|I\| < \|C\|$ (number of candidate input attributes) **do**
Step 2.1	**For each** candidate input attribute $A_{i'}$ /$A_{i'} \in C; A_{i'} \notin I$ **do** **If** $A_{i'}$ is continuous **then** Return the best threshold splits of $A_{i'}$. Return statistically significant conditional mutual information *cond_MI*$_{i'}$ between $A_{i'}$ and the target attribute A_i. **End Do**
Step 2.2	Find the candidate input attribute $A_{i'}$* maximizing *cond_MI*$_{i'}$
Step 2.3	If *cond_MI*$_{i'*}$ = 0, then **End Do**. Else Expand the network by a new hidden layer associated with the attribute $A_{i'}$, and add $A_{i'}$ to the set *I* of selected input attributes: $I = I \cap A_{i'}$.
Step 2.4	**End Do**
Step 3	Return the set of selected input attributes *I* and the network structure

The algorithm calculates the conditional mutual information of a candidate input attribute $A_{i'}$ and a target attribute A_i, given a node *z,* by the following formula (based on Cover, 1991):

$$MI\ (A_{i'}; A_i / z) = \sum_{j=0}^{M_i - 1} \sum_{j'=0}^{M_{i'} - 1} P\ (V_{ij}; V_{i'j'}; z) \bullet \log \frac{P\ (V_{i'j'}^{ij} / z)}{P\ (V_{i'j'} / z) \bullet P\ (V_{ij} / z)} \quad (1)$$

where

M_i / $M_{i'}$ - number of distinct values of the target attribute A_i /candidate input attribute $A_{i'}$ respectively.

$P\ (V_{i'j'}/ z)$ - an estimated conditional (*a posteriori*) probability of a value *j'* of the candidate input attribute $A_{i'}$ given the node *z* (also called a *relative frequency estimator*)

$P\ (V_{ij}/ z)$ - an estimated conditional (*a posteriori*) probability of a value *j* of the target attribute A_i given the node *z*.

$P\ (V_{i'j'}^{ij}/ z)$ - an estimated conditional (*a posteriori*) probability of a value *j'* of the candidate input attribute $A_{i'}$ and a value *j* of the target attribute A_i given the node *z*.

$P\ (V_{ij}; V_{i'j'}; z)$ - an estimated joint probability of a value *j* of the target attribute A_i, a value *j'* of the candidate input attribute $A_{i'}$, and the node *z*.

The statistical significance of the estimated conditional mutual information, is evaluated by using the likelihood-ratio statistic (based on Attneave, 1959):

$$G^2\ (A_{i'} ; A_i / z) = 2 \bullet (\ln 2) \bullet E^*(z) \bullet MI\ (A_{i'} ; A_i / z) \quad (2)$$

Where $E^*(z)$ is the number of records associated with the node *z*

The Likelihood-Ratio Test (see Rao and Toutenburg, 1995) is a general-purpose method for testing the null hypothesis H_0 that two discrete random variables are statistically independent. For example, if the customer credibility is independent of his/her other investments in the bank, the proportion of credible customers among those having other investments should be equal to their proportion among those who do not. The Likelihood-Ratio Test is directly related to the information theory, since independence of two attributes implies that their expected mutual information is zero. If H_0 holds, then the likelihood-ratio test statistic $G^2 (A_{i'}; A_i / z)$ is distributed as chi-square with $(NI_{i'} (z) - 1) \bullet (NT_i (z) - 1)$ degrees of freedom, where $NI_{i'} (z)$ is the number of distinct values of a candidate input attribute $A_{i'}$ at node z and $NT_i (z)$ is the number of distinct values of the target attribute A_i at node z. The default significance level (*p-value*) used by the information-theoretic algorithm is 0.1%. We have found empirically that in most datasets, higher values of the *p-value* tend to decrease the generalization performance of the network.

2.3 Computational Complexity

The computational complexity of the network construction procedure is calculated by using the following notation:

n - total number of instances in a training data set

$|C|$ - total number of candidate input attributes

$m = |I|$ -number of hidden layers (input attributes), $m \leq |C|$

$p = m/|C|$ - portion of input attributes selected by the algorithm

M_T - maximum domain size of a target attribute (maximum number of classes)

The computational "bottleneck" of the algorithm is calculating the estimated conditional mutual information between every binary partition of a candidate input attribute and a target attribute, given a hidden node. Since each node of m-th hidden layer represents a conjunction of values of m input attributes, the total number of nodes at a layer m is apparently bounded by $(M_c)^m$. However, we restrict defining a new node by the requirement that there is at least one instance associated with it. Thus, the total number of nodes at any hidden layer cannot exceed the total number of instances (n). In most cases, the number of nodes will be much smaller than n, due to instances having identical values and the statistical significance requirement of the likelihood-ratio test.

The calculation of the conditional mutual information is performed at each hidden layer of the information-theoretic network for all candidate input attributes at that layer. The number of possible partitions of a continuous attribute is bounded by $n\log_2 n$ (Fayyad and Irani, 1993). For every possible partition, the conditional information is summarized over all nodes of the final layer. This implies that the total number of calculations is bounded by:

$$n \bullet n \bullet \log_2 n \bullet M_T \bullet \sum_{m=0}^{p|C|} (|C| - m) \le \frac{n^2 \bullet \log_2 n \bullet M_T \bullet |C|^2 \bullet p \bullet (2-p)}{2} \quad (3)$$

The actual number of calculations will be usually much smaller than this bound, since the number of tested partitions may be less than the number of distinct values (resulting from the likelihood-ratio test). The number of distinct values, in turn, may be much lower than the total number of records *(n).* Thus, the run time of the search procedure is quadratic-logarithmic in the number of records and quadratic polynomial in the number of initial candidate input attributes. Moreover, it is reduced by the factor of $p(2-p)$.

2.4 Prediction with IFN

The predicted value (class) of the target attribute A_i in a new instance can be found by traversing an info-fuzzy network in the following way:

Step 1 - Start with the root node ($z = 0$) and an empty set of input attributes ($m = 0$).

Step 2 - If the node is a terminal node, go to *Step 5*. Otherwise, go to next step.

Step 3 - Increment the number of input attributes ($m = m+1$) and calculate the next hidden node z by the value of the input attribute m in an instance.

Step 4 - Go to Step 2.

Step 5 - Get the predicted value j^* of the target attribute A_i at the node z by the *maximum a posteriori* rule: $j^* = \arg\max_j \{P(V_{ij} / z)\}$, where $P(V_{ij}/z)$ is an estimated conditional probability of a value j of the target attribute A_i, given the node z.

As shown above, IFN can be used to predict values (classes) of target attributes in a manner, similar to the approach of other decision-tree methods.

2.5 Comparison to Related Methods

Though traditional methods of neural network training, like Back-Propagation, are aimed at determining the values of the connection weights in a given network, there are some techniques to dynamically modify the network structure itself. The Cascade-Correlation algorithm (Fahlman and Lebiere, 1991) is one such technique. Cascade-Correlation starts with a minimal network having only input and output units (nodes) and it creates a multi-layer structure by adding single-unit layers to the network one by one. The network construction continues until the error is acceptably small or the maximum number of iterations is exceeded. Increasing the number of units in each layer can reduce the depth of a cascade-correlation network (see Phatak and Koren, 1994). The main differences between Cascade-Correlation and IFN include the model architecture (the layers of a neural network are not associated with specific features) and the lack of any statistical significance tests in the Cascade-Correlation methods.

Cios and Liu (1992) have proposed an algorithm, called Continuous ID3 (CID3), which combines the concepts of neural network architecture and decision-tree learning. The CID3 algorithm incrementally generates a multi-layer network, where each hidden layer is associated with a decision tree grown by the ID3 algorithm (Quinlan, 1986). ID3 is trained on *extracted features* calculated as linear combinations of the original features (hyperplanes). Each new node in a hidden layer represents an extracted feature. The algorithm finds the combination weights that minimize the information entropy at a given level of a decision tree. To further reduce the entropy, a new node is added to the same layer and or a new layer is added to the network. The network construction stops, when the entropy converges to zero. Unlike IFN, CID3 is limited to continuous input attributes. Since CID3 is using a linear combination of all input attributes, it does not reduce directly the number of original features. In addition, CID3 does not test the statistical significance of the entropy values.

The idea of using a restricted set of features in each layer of a decision tree is implemented in the *conditional cluster tree* constructed by the Conditional Rule Generation (CRG) algorithm (Bischof and Caelli, 1994). The algorithm generates rules, which satisfy domain-specific compatibility constraints that are completely ignored by fully automated learning methods. The IFN method also restricts the features to be used in each layer of the network, but, unlike CRG, IFN does not use any domain knowledge and it selects the relevant features automatically.

The most popular methods for automated construction of decision trees include ID3 (Quinlan, 1986), CART (Breiman et al., 1984), and C4.5 (Quinlan, 1993). All these algorithms grow a tree by recursive partitioning of a subset of training instances. Consequently, the features used by different generated rules, as well as their ordering, may be completely different. The only exception is the EODG algorithm (Kohavi and Li, 1995), which requires all the nodes of a given layer to be split on the same feature. Table 2 compares between the IFN construction procedure, the "classical" decision-tree methods (CART and C4.5), and the EODG algorithm. According to the table, the unique features of IFN include the use of conditional mutual information as a feature selection criterion in each layer, multi-way splits of continuous attributes, and pre-pruning by the likelihood-ratio test.

Table 2 Comparison of Decision-Tree Algorithms

Property	**CART / C4.5**	**EODG**	**IFN**
Tree construction strategy	Recursive partitioning of a subset of training instances at each node	Repetitive partitioning of all training instances in every level	Repetitive partitioning of all training instances in every layer (except for instances at unsplit nodes)
Feature selection	The best feature is selected for every node	All nodes in a given level are split on the same feature	All nodes in a given layer are split on the same feature
Splitting criteria	CART: Gini, Twoing, Entropy C4.5: Gain Ratio	Adjusted Mutual Information	Conditional Mutual Information
Splits on continuous features	Binary (threshold) splits only The same feature may be tested at different levels	Binary (threshold) splits only The same feature may be tested at different levels	Multi-way splits The same feature is not tested at more than one layer
Pre-pruning criteria	Minimum number of cases for each outcome at a node	The instances are split on all features	Likelihood-Ratio Test
Post-pruning criteria	CART: cost-complexity pruning C4.5: Reduced error bottom-up pruning	Bottom-up error-based pruning Top-down merging of nodes	No post-pruning
Target layer	No	Yes	Yes

Turney (1995) proposes three main ways to improve stability of a given learner, namely: increasing the number of training examples, increasing the strength of the algorithm bias, and memorizing previously learned concepts. If the size of the training set is limited, applying a bias is necessary for maintaining the algorithm's stability. The IFN methodology is an attempt to implement a stable learning algorithm by introducing an *exclusive*, or *restriction* bias (see Mitchell, 1997) against decision trees that use different attributes at the nodes of the same level. In the next section, we evaluate the *correctness* of the IFN algorithm vs. other decision-tree methods in terms of accuracy, stability, and complexity.

3. Empirical Results

A *meta-learning* approach, called bagging (see Breiman, 1996) can improve the accuracy of unstable methods, like neural networks and decision trees. However, the resulting model is hardly comprehensible and, thus, cannot be used for efficient knowledge extraction. Domingos (1997a and 1998) has developed a method, called CMM, for simplifying the output of meta-learners. Domingos is using the *output size* of the extracted rule set as a measure of model description length. The output size is calculated by counting one unit for each antecedent and each consequent of every rule. In his paper, Domingos has shown on 26 representative datasets that CMM's complexity is usually a small multiple of C4.5 RULES (2-6), while it produces more stable and accurate results. Stability was evaluated by the following semantic measure of agreement based on (Turney, 1995):

$$Stab = 100\% \times \frac{1}{ne} \sum_{e=1}^{ne} \left[\frac{2}{nr(nr-1)} \sum_{i=1}^{nr} \sum_{j=1}^{i-1} agree_{eij} \right] \qquad (4)$$

where *ne* is the number of testing examples randomly generated from the original instance space, *nr* is the number of train-test runs (each producing a classification model from a subset of the original dataset), and $agree_{eij}$ is 1 if models *i* and *j* predicted the same class for a testing example *e*, and 0 otherwise. Intuitively, *Stab* represents the percentage of pairwise agreements between the models over a set of testing examples.

Table 3 Empirical Results: Predictive accuracy

			Attributes			**Predictive Accuracy**			
Dataset	**Records**	**Classes**	**Continuous**	**Nominal**	**Total**	**CMM**	**Bagging**	**C4.5RULES**	**IFN**
Credit	690	2	6	8	14	87.5±1.1	88.3±0.8	86.2±1.1	84.1±1.0
Diabetes	768	2	8	0	8	75.5±0.9	76.4±0.9	73.6±0.8	72.2±1.0
Glass	214	6	9	0	9	73.1±1.5	77.4±1.5	71.7±1.7	60.9±1.0
Heart	270	2	6	7	13	81.8±1.5	80.2±1.7	78.0±1.7	75.8±1.1
Iris	150	3	4	0	4	94.3±1.1	94.3±1.1	94.7±1.1	95.6±0.3
Liver	345	2	6	0	6	66.0±1.8	69.6±1.9	63.7±1.4	63.7±1.7
Lung -cancer	32	3	0	57	57	40.0±7.5	36.7±7.2	31.7±7.0	31.6±3.9
Wine	178	3	13	0	13	94.2±1.0	95.8±1.0	94.7±1.6	90.4±0.9
Average	**330.9**	**2.9**	**6.5**	**9.0**	**15.5**	**76.55**	**77.34**	**74.29**	**71.79**

IFN has been applied to eight datasets, chosen randomly from Table 2 in (Domingos, 1998). The left part of Table 3 in this paper shows the dimensionality of each dataset. The datasets included in our experiments vary in the number of cases, the number and type of original features, and the number of target classes. The right part of Table 3 compares the predictive accuracy of IFN to the methods covered by Domingos' paper (CMM, Bagging, and C4.5RULES). To estimate IFN predictive accuracy , we have performed 10 runs of 10-fold cross-validation, each based on a different random partitioning of the data set. The confidence intervals of the mean predictive accuracy have been calculated for the 0.95 confidence level, using *t*-distribution with *n-1* degrees of freedom, where *n* is the number of 10-fold cross-validation runs (10). As expected, the bias of IFN towards single-feature layers affects its predictive accuracy. The IFN's average accuracy is lower than CMM's accuracy by almost 5%. Only in one dataset out of eight (Iris), the IFN predictive accuracy is significantly higher than the average accuracy of the three other methods. As indicated several times in this paper, accuracy is not the only objective of learning from data. The performance of the algorithms with respect to two other measures (complexity and stability) is evaluated below.

Table 4 Empirical Results: Output Size and Stability (- 99% significant difference vs. CMM)**

	Output Size				**Stability**				
Dataset	**CMM**	**Bagging**	**C4.5RULES**	**IFN**	**CMM**	**Bagging**	**C4.5RULES**	**IFN**	
Credit	104.9	2181.2	49.4	13	89.7%	90.6%	81.4%	92.5%	**
Diabetes	82.4	3476.6	38.9	31	83.1%	85.4%	76.3%	87.1%	**
Glass	214.6	1740.5	61.8	40	67.1%	70.5%	69.3%	78.2%	**
Heart	177.7	1396.9	48.5	35	80.2%	88.8%	75.1%	87.3%	**
Iris	17.6	303.5	11	6	96.0%	96.9%	99.2%	97.4%	**
Liver	116.1	2329.9	53	13	82.1%	85.0%	70.6%	85.6%	**
Lung-cancer	198.8	255.2	9.7	4	55.0%	58.5%	52.9%	54.4%	
Wine	69.4	399.1	15	21	71.1%	79.4%	74.6%	96.7%	**
Average	**122.69**	**1510.36**	**35.91**	**20.38**	**78.0%**	**81.9%**	**74.9%**	**84.9%**	

The left part of Table 4 above shows the output complexity of different methods. IFN provides the smallest average output size (20 only vs. 123 for CMM and 36 for C4.5Rules), which, as shown above, is accompanied by a small loss of accuracy (about 5% vs. CMM). A 12% difference in predictive accuracy (the Glass dataset) may justify a four times increase in the output size, but an improvement of about 3% only (Credit) can hardly support an eight times increase in the model complexity. One should also remember that CMM is a time-consuming algorithm, which requires generation of multiple models based on original and artificially created data, while IFN is constructing a single model from the original data. Unfortunately, (Domingos, 1998) does not present the actual run times of the CMM algorithm.

Semantic stability of the algorithms, based on (Turney, 1995), is presented in the right part of Table 4. According to the obtained results, IFN is clearly the most stable method out of the four algorithms evaluated. In seven datasets out of eight, IFN is more stable than CMM at the significance level of 99% and higher. Only in one dataset (Lung-Cancer) IFN is slightly less stable than the CMM method. Thus, most IFN models are smaller and more stable than the models produced by three alternative methods, while maintaining a reasonable level of predictive accuracy.

The case of the famous Iris dataset is particularly interesting: IFN predictive accuracy appears to be slightly higher than the accuracy of the other three methods. At the same time, IFN provides a smaller output size (six nodes only) and higher stability than the multiple-model methods (bagging and CMM). Therefore, we present in Figure 2 the network induced from this data set. The associated set of prediction rules is shown below:

Rule No. 1	If Petal Length is between 1.0 and 3.0 then Class is 1
Rule No. 2	If Petal Length is between 3.0 and 4.8 then Class is 2
Rule No. 3	If Petal Length is more than 4.8 then Class is 3

As one can see from the network and the list of rules, the IFN method has left only one predicting attribute in the model (Petal Length) vs. two attributes (Petal Length and Petal Width) used by other methods like C4.5.

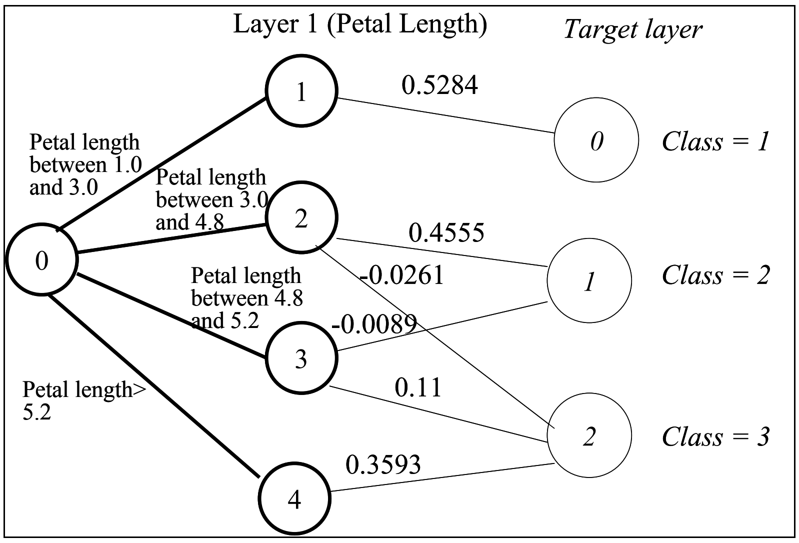

Figure 2 IFN Example 2: Iris Dataset

4. Conclusions

Decision trees are known as highly efficient tools of machine learning and data mining, capable to produce accurate and easy-to-understand models. Poor stability is the "Achilles heel" of decision-tree methods. The existing, computationally intensive approaches to the instability problem are based on combining multiple learners, which often comes at the cost of losing the model comprehensibility. This paper describes a single-model method, called IFN, for building semantically stable and compact decision trees. The proposed method is based on information-theoretic selection of predicting features and statistical significance testing as a method of model pre-pruning. Experimental results show IFN to be a promising approach to learning patterns from data. Improving the accuracy of the IFN models, while preserving their compact description length, is a subject for future research. Another important issue is defining and evaluating syntactic, topology-related measures of decision tree stability.

References

F. Attneave, Applications of Information Theory to Psychology, Holt, Rinehart and Winston , 1959.

W. Bischof and T. Caelli, Learning Structural Descriptions of Patterns: A New Technique for Conditional Clustering and Rule Generation, Pattern Recognition, Vol. 27, No. 5, pp. 689-697, 1994.

C.L. Blake & C.J. Merz, UCI Repository of machine learning databases [http://www.ics.uci.edu/~mlearn/MLRepository.html], Department of Information and Computer Science, University of California at Irvine, Irvine, CA, 1998.

L. Breiman, J.H. Friedman, R.A. Olshen, & P.J. Stone, Classification and Regression Trees, Wadsworth, Belmont, CA, 1984.

L. Breiman, Bagging Predictors, Machine Learning, vol. 24, pp. 123-140, 1996.

K.J. Cios and N. Liu, A Machine Learning Method for Generation of a Neural Network Architecture: A Continuous ID3 Algorithm, IEEE Transactions on Neural Networks, Vol. 3, No. 2, pp. 280-291, 1992.

T. M. Cover, Elements of Information Theory, Wiley, New York, 1991.

P. Domingos, Knowledge Acquisition from Examples via Multiple Models., Proc. Fourteenth International Conference on Machine Learning, Nashville, TN, pp. 98-106, 1997a.

P. Domingos, Why Does Bagging Work? A Bayesian Account and its Implications, Proc. the Third International Conference on Knowledge Discovery and Data Mining, Newport Beach, CA, pp. 155-158, 1997b.

P. Domingos, Knowledge Discovery via Multiple Models, Intelligent Data Analysis, No. 2, pp. 187-202, 1998.

S. E. Fahlman and C. Lebiere, The Cascade-Correlation Learning Architecture, Internal Report CMU-CS-90-100, Carnegie Mellon University, 1991.

U. Fayyad and K. Irani, Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning, Proc. Thirteens International Joint Conference on Artificial Intelligence, San Mateo,1993.

U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, From Data Mining to Knowledge Discovery: An Overview. In Advances in Knowledge Discovery and Data Mining, U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, Eds. AAAI/MIT Press, Menlo Park, CA, pp. 1-30, 1996.

Y. Freund & R. E. Schapire, Experiments with a new boosting algorithm. Proc. Thirteenth International Conference on Machine Learning, pp. 148--156, Bari, Italy: Morgan Kaufmann, 1996.

P. Kilpeläinen, Tree Matching Problems with Applications to Structured Text Databases, Ph.D. Dissertation, University of Helsinki, 1992.

R. Kohavi and C-H. Li, Oblivious Decision Trees, Graphs, and Top-Down Pruning, Proc. of International Joint Conference on Artificial Intelligence (IJCAI), pages 1071-1077, 1995.

H. Liu and H. Motoda, Feature Selection for Knowledge Discovery and Data Mining, Kluwer, Boston, 1998.

O. Maimon and M. Last, Knowledge Discovery and Data Mining, The Info-Fuzzy Network (IFN) Methodology, Kluwer Academic Publishers, 2000.

T.M. Mitchell, Machine Learning, McGraw-Hill, New York, 1997.

M. Pelillo, K. Siddiqi, and S.W. Zucker, Matching Hierarchical Structures Using Association Graphs, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 21, No. 11, pp. 1105-1120, 1999.

D.S. Phatak and I. Koren, Connectivity and Performance Tradeoffs in the Cascade Correlation Learning Architecture, IEEE Transactions on Neural Networks, Vol. 5, No. 6, pp. 930-935, 1994.

J.R. Quinlan, Induction of Decision Trees, Machine Learning, vol. 1, no. 1, pp. 81-106, 1986.

J. R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann, San Mateo, CA, 1993.

C.R. Rao and H. Toutenburg, Linear Models: Least Squares and Alternatives, Springer-Verlag, Berlin, 1995.

P. Turney, Bias and the Quantification of Stability, Machine Learning, no. 20, pp. 23-33, 1995.