Appears in the European Conference on Machine Learning (ECML), 1995

The Power of Decision Tables

Ron Kohavi

Computer Science Department
Stanford University
Stanford, CA. 94305
ronnyk@CS.Stanford.EDU
http://robotics.stanford.edu/~ronnyk

Abstract. We evaluate the power of decision tables as a hypothesis space for supervised learning algorithms. Decision tables are one of the simplest hypothesis spaces possible, and usually they are easy to understand. Experimental results show that on artificial and real-world domains containing only discrete features, IDTM, an algorithm inducing decision tables, can sometimes outperform state-of-the-art algorithms such as C4.5. Surprisingly, performance is quite good on some datasets with continuous features, indicating that many datasets used in machine learning either do not require these features, or that these features have few values. We also describe an incremental method for performing cross-validation that is applicable to incremental learning algorithms including IDTM. Using incremental cross-validation, it is possible to cross-validate a given dataset and IDTM in time that is linear in the number of instances, the number of features, and the number of label values. The time for incremental cross-validation is independent of the number of folds chosen, hence leave-one-out cross-validation and ten-fold cross-validation take the same time.

1 Introduction

Write the vision, and make it plain upon tables,
that he may run that readeth it.
—Habakkuk 2:2

Given a dataset of labelled instances, supervised machine learning algorithms seek a hypothesis that will correctly predict the class of future unlabelled instances. In the machine learning literature, many representations for hypotheses have been suggested, including decision trees, decision graphs, neural networks, k-DNF formulae, automata, Lisp programs, and probability measures.

We investigate the power of one of the simplest representations possible—a decision table with a default rule mapping to the majority class. This representation, called **DTM** (Decision Table Majority), has two components: a **schema** which is a set of features that are included in the table, and a **body** consisting of labelled instances from the space defined by the features in the schema. Given an unlabelled instance, a decision table classifier searches for exact matches in the decision table using only the features in the schema (note that there may be many matching instances in the table). If no instances are found, the majority class of the DTM is returned; otherwise, the majority class of all matching instances is returned.

To build a DTM, the induction algorithm must decide which features to include in the schema and which instances to store in the body. In this paper, we restrict ourselves to the former problem, called feature subset selection.

Given a target function f and a hypothesis class $\mathcal{H}$, we define the **optimal features** to be the features used in a hypothesis h in $\mathcal{H}$ that has the highest future prediction accuracy with respect to f. Because the hypothesis space is limited in its expressive power, the optimal features may not include all relevant features.

To search for the optimal features, the wrapper model (John, Kohavi & Pfleger 1994) is used. In the wrapper model, the induction algorithm is used as a black box, and a search through the space of feature subsets is made by a "wrapper" algorithm. In this paper, we search using best-first search and estimate the future prediction accuracy (the heuristic required for the best first search) with ***k*-fold cross-validation**.

The goal of this paper is to evaluate the representation power of DTMs. In experiments with feature subset selection for decision tree algorithms, we have observed that in many cases the decision-trees were nearly complete, *i.e.*, the leaves represented almost all combinations of the chosen subset of the features. We then conjectured that a simple decision table on a subset of the features might be a good hypothesis space. While we use a specific technique for selecting the features—the wrapper model—our aim is not to show that the specific method for selecting features is good, but rather to show that at least one method for selecting the schema works well. It is conceivable that other methods, perhaps better and faster, exist.

The chances of getting a perfect match on the values of continuous features are slim: even a single truly continuous feature in the schema will make the table useless. Our initial experiments were therefore restricted to datasets containing only discrete features. To determine how weak the performance of IDTM is on datasets with continuous features, we also report on such experiments. Surprisingly, performance is not significantly worse than that of C4.5 (Quinlan 1993) in some cases. On those that performance is not significantly worse than C4.5, the algorithm ignores the continuous features or uses those features that have few values.

The paper is organized as follows. Section 2 formally defines DTMs and the problem of finding an optimal feature subset. Section 3 describes how we search for the optimal feature subsets using best-first search to guide the search and cross-validation to estimate the accuracy. Section 4 details the experimental methodology and the results. Section 5 describes related work on decision tables and feature subset selection. Section 6 concludes with a summary and directions for future work.

2 Decision Tables and Optimal Features Subsets

Given a training sample containing labelled instances, an induction algorithm builds a hypothesis in some representation. The representation we investigate here is a decision table with a default rule mapping to the majority class, which we abbreviate as **DTM**. A DTM has two components:

1. A **schema,** which is a set of features.
2. A **body,** which is a multiset of labelled instances. Each instance consists of a value for each of the features in the schema and a value for the label.

Given an unlabelled instance I, the label assigned to the instance by a DTM classifier is computed as follows. Let $\mathcal{I}$ be the set of labelled instances in the DTM exactly matching the given instance I, where only the features in the schema are required to match and all other features are ignored. If $\mathcal{I} = \emptyset$, return the majority class in the DTM; otherwise, return the majority class in $\mathcal{I}$. Unknown values are treated as distinct values in the matching process.

Let $\mathbf{err}(\boldsymbol{h}, \boldsymbol{f})$ denote the error of a hypothesis h for a given target function f. Since f is never known for real-world problems, we estimate the error using an independent test set $\mathcal{T}$ as

$$\widehat{\mathrm{err}}(h, \mathcal{T}) = \frac{1}{|\mathcal{T}|} \sum_{(x_i, y_i) \in \mathcal{T}} L(h(x_i), y_i) \ ,$$

where L is a loss function. In the rest of the paper we assume a zero-one loss function, *i.e.*, zero if $h(x) = y$ and one otherwise. The **approximate accuracy** is defined as $1 - \widehat{\mathrm{err}}(h, \mathcal{T})$.

An **optimal feature subset,** $\boldsymbol{\mathcal{A}}^*$**,** for a given hypothesis space $\mathcal{H}$ and a target function f is a subset of the features $\mathcal{A}^*$ such that there exists a hypothesis h in $\mathcal{H}$ using only features in $\mathcal{A}^*$ and having the lowest possible error with respect to the target function f. (Note that the subset need not be unique.) As the following example shows, relevant features are not necessarily included in the optimal subset.

Example 1.
Let the universe of possible instances be $\{0, 1\}^3$, that is, three Boolean features, say X_1, X_2, X_3. Let the distribution over the universe be uniform, and assume the target concept is $f(X_1, X_2, X_3) = (X_1 \wedge X_2) \vee X_3$. Under any reasonable definition of relevance, all variables are relevant to this target function.

If the hypothesis space is the space of monomials, *i.e.*, conjunctions of literals, the only optimal feature subset is $\{X_3\}$ The accuracy of the the monomial X_3 is 87.5%, the highest accuracy achievable within this hypothesis space. ▮

An induction algorithm using DTMs as the underlying hypothesis space must decide which instances to store in the table and which features to include in the schema. In this paper we assume the induction algorithm includes the projections of all instances defined by the schema in the DTM, but we do not restrict the subset of features to use in the schema in any way. Let $\mathcal{A} = \{X_1, \ldots, X_n\}$ be a set of features and let $\mathcal{S}$ be a sample of $\boldsymbol{m}$ instances over the features in $\mathcal{A}$. Given a subset of features $\mathcal{A}' \subseteq A$, $\boldsymbol{DTM(\mathcal{A}', \mathcal{S})}$ is the DTM with schema $\mathcal{A}'$ and a body consisting of all instances in $\mathcal{S}$ projected on $\mathcal{A}'$. The goal of the induction algorithm is to chose a schema $\mathcal{A}^*$ such that

$$\mathcal{A}^* = \operatorname*{arg\,min}_{\mathcal{A}' \subseteq \mathcal{A}} \operatorname{err}(DTM(\mathcal{A}', \mathcal{S}), f) \ . \tag{1}$$

The schema $\mathcal{A}^*$ consists of an optimal feature subset for a DTM under the assumption that all instances from the training set are stored in the body of the decision table.

3 Finding an Optimal Feature Subset

In this section, we describe **IDTM** (Inducer of DTMs), an induction algorithm that induces DTMs. The goal of IDTM is clear: find the feature subset $\mathcal{A}^*$ that is described in Equation 1. Since the target function f is unknown, no learning algorithm can compute the exact error: it can only be approximated. Furthermore, the number of feature subsets for n features is 2^n, a space too large to search exhaustively even for moderately sized n.

An interesting way to view the induction process is to think of the feature subset selection algorithm as wrapping around a trivial induction algorithm that simply creates a DTM from the full dataset it receives (see John et al. (1994) for details on the wrapper approach). The wrapper is the only part that makes the inductive leap of which features to use.[1]

We now give an overview of the feature subset selection mechanism, which transforms the problem into one of state space search with probabilistic estimates; further details can be found in Kohavi (1994*c*).

3.1 Searching the Space of Feature Subsets

In order to search the space of feature subsets effectively, we transform the problem into a state space search and use best-first search to heuristically search the space (Ginsberg 1993, Nilsson 1980).

The states in the space are feature subsets; operators can add or delete a feature; the initial node can be either the set of all features or the empty set; and the evaluation function is cross-validation (described below). Since we are aiming for the optimal feature subset, there is no goal node. The optimization

[1] Holte (personal communication) remarked that this type of learning is basically a preprocessing step (feature selection), and an optional post-processing step to simplify the rules, with *nothing* in between.

problem requires a termination condition, and the algorithm we used stops after a fixed number of node expansions do not yield a node with a better estimated accuracy than the current best estimate.

To estimate future prediction accuracy, cross-validation, a standard accuracy estimation technique (Weiss & Kulikowski 1991, Breiman, Friedman, Olshen & Stone 1984, Stone 1974), is used. Given an induction algorithm and a dataset, k-fold cross-validation splits the data into k approximately equally sized partitions, or folds. The induction algorithm is executed k times; each time it is trained on $k-1$ folds and the generated hypothesis is tested on the unseen fold, which serves as a test set. The estimated accuracy is computed as the average accuracy over the k test sets.

The estimated accuracy for each cross-validation fold is a random variable that depends on the random partitioning of the data. We observed high variance in the accuracy estimates and ameliorate this disturbing phenomenon by repeating the cross-validation t times. Following John's suggestion (1994), we used a 10% trimmed mean of the kt folds.

3.2 Incremental Cross-Validation

Each feature subset, represented as a node in the state space, is evaluated by cross-validation. One of the main problems with regular k-fold cross-validation is that the algorithm is run k times, introducing a multiplicative factor of k in the running time. We now explain how to speed up cross-validation time for algorithms that support incremental addition and deletion of instances. We feel that this digression is important because the simple idea of incremental cross-validation is what makes the IDTM algorithm practical.

The idea in incremental cross-validation is that instead of training k times on $k-1$ folds each time, we train on the full dataset, then delete the instances in one fold, test on that fold, and insert the instances back. The delete-test-insert phase is repeated for each of the k folds. If the algorithm is guaranteed to produce the same results in incremental mode as in batch mode, this incremental version of cross-validation is guaranteed to produce the exact same result as batch cross-validation.

Proposition 1 Incremental Cross-Validation.
The running time of incremental cross-validation is

$$O(T + m(t_d + t_c + t_i)) \ ,$$

where T is the running time of the induction algorithm on the full dataset, m is the number of instances, and t_d, t_c, and t_i represent the time it takes to delete an instance, classify an instance, and insert an instance, respectively.

Proof: Incremental cross-validation starts out by running the original induction algorithm on the full dataset. Since each instance appears in exactly one fold, it is deleted once, classified once, and inserted once during the overall incremental cross-validation phase. ∎

Example 2.
Conducting k-fold cross-validating of a decision tree induction algorithm and a dataset is deemed an expensive operation because one typically builds k decision trees from scratch, one for each fold. However, Utgoff (1994) shows how to incrementally add and delete instances in a way that is guaranteed to generate the same tree as a batch algorithm. Thus, one can incrementally cross-validate decision trees much faster.

Nearest neighbor algorithms support incremental addition and deletion of instances by simply adding and removing prototype points. Since these operations are fast, it can be shown that incremental cross-validation of a dataset with m instances and n features with a simple nearest neighbor induction algorithm takes $O(m(m \cdot n))$ time; incremental cross-validation of a weighted regression nearest neighbor takes $O(m(m \cdot n^2 + m^3))$ time as shown in Moore & Lee (1994), Maron & Moore (1994), and Moore, Hill & Johnson (1992).

We now describe the data structures that allow fast incremental operations on DTMs. The underlying data structure that we use is a universal hash table (Cormen, Leiserson & Rivest 1990). The time to compute the hash function is $O(n')$ where n' is the number of feature values in the DTM's schema, and the expected lookup time (given the hashed value of the instance) is $O(1)$ if all objects (unlabelled instances) stored are unique. To ensure that all stored objects are unique, we store with each unlabelled instance ℓ counter values, where ℓ is the number of label values. Each counter value c_i represents the number of instances in the training set having the same underlying unlabelled instance and label l_i.

To classify an instance, the unlabelled instance is found in the hash table and the label matching the highest counter value is returned.[2] The overall expected time to classify an instance is thus $O(n' + \ell)$.

To delete an instance, the underlying unlabelled instance is found and the appropriate label counter is decreased by one; if all counters are zero, the underlying unlabelled instance is deleted from the table. Inserting instances is done similarly. Class counts must be kept for the whole body of the DTM in order to change the majority class.

Corollary 2 Incremental Cross-Validation of IDTMs.
The overall time to cross-validate an IDTM with n' features in the schema and a dataset with m instances and ℓ label values is $O(m(n' + \ell))$.

Proof: All DTM operations have time complexity $t_d = t_c = t_i = O(n' + \ell)$. The overall time to build a DTM from scratch is the same as m insertions; thus by Proposition 2, the overall time for the cross-validation $O(m(n' + \ell))$. ∎

[2] The running time could further decreased to $O(n')$ by computing the majority of every unlabelled instance in advance, but the counters are needed for the incremental operations.

3.3 Choosing the Number of Folds

The time to incrementally cross-validate an IDTM and a dataset for any number of folds is the same. Leave-one-out is almost unbiased (Efron 1983) and was commonly considered the preferred method for cross-validation. Recently Zhang (1992) and Shao (1993) proved that, for linear models, using leave-one-out cross-validation for model selection is asymptotically inconsistent in the sense that the probability of selecting the model with the best predictive power does not converge to 1 as the total number of observations approaches infinity. The theorems show that in order to select the correct model, as the number of instances in the dataset grows, the number of instances left out for testing should grow as well. Zhang showed that the models chosen by any k-fold cross-validation for any k will overfit in the sense that too many features will be selected. However, for moderately sized data sets, he claimed that 10 to 15 folds are reasonable choices.

Empirically, we have observed similar results, namely that using ten-fold cross-validation is slightly better than leave-one-out. Similar observations were made by Weiss (1991). While the differences are usually small, especially for feature subset selection where only the relative ranking of different subsets matters, there is one extreme case that deserves special mention: the Monk1 dataset.

Example 3 Leave-one-out on Monk1.
The Monk1 problem (Thrun etal. 1991) has a standard training and test set. There are no duplicate instances, nor is there noise in the training set. A DTM with a schema that has all the features and that is tested on a test set disjoint from the training set always predicts the majority class; hence it is equivalent to an induction algorithm that predicts a constant function—*True* or *False*—depending on the prevalent class in the training set.

The estimated accuracy using leave-one-out cross-validation on a DTM with all the features (or equivalently, a majority inducer) and the standard training set for the Monk1 problem is 0.0%! The example shows an inherent problem with cross-validation that applies to more than just a majority inducer. In a no-information dataset where the label values are completely random, the best an induction algorithm can do is predict majority. Leave-one-out on such a dataset with 50% of the labels for each class and a majority inducer would still predict 0% accuracy. ∎

The reason for this phenomenon is that the standard training set for the Monk1 problem has 62 positive instances and 62 negative instances. Each time an instance is removed in leave-one-out, the other class is the more prevalent in the training set and the majority inducer predicts the wrong label for the test instance.

We have observed a similar phenomenon even with ten-fold CV. The iris dataset has 150 instances, 50 of each class. Predicting any class would yield 33.3% accuracy, but ten-fold CV using a majority induction algorithm yields

21.5% accuracy (averaged over 100 runs of ten-fold CV). The reason is that if there is a majority of one class in the training set, there is a minority of that class in the test set. (See Schaffer (1994) for a discussion along these lines.)

4 Experiments with IDTM

We now describe experiments conducted with IDTM, the induction algorithm for DTMs. The experiments were done on all the datasets at the UC Irvine repository (Murphy & Aha 1994) and StatLog repository (Taylor, Michie & Spiegalhalter 1994) that contain only discrete features. To test the performance on datasets with continuous features, we chose the rest of the StatLog datasets except shuttle, which was too big, and all the datasets used by Holte (1993).

4.1 Methodology

We now define the exact settings used in the algorithms. The estimated accuracy for each node was computed using ten-fold cross validation. Because of the high variability of the estimates, the cross-validation was repeated (shuffling the instances between runs) until the standard deviation of the mean went below one percent or until ten cross-validations runs have been executed.

The termination condition for the search was a consecutive sequence of five node expansions that did not generate a feature subset with an estimated accuracy of at least 0.1% better than the previous best subset.

For datasets with a specified training and test set, we executed the algorithm once. For the rest of the datasets, we performed ten-fold cross-validation around IDTM.[3] In the comparisons with C4.5, the same cross-validation folds were used for both algorithms.

As in Holte's work, we believe that the weakest part of the IDTM algorithm is the accuracy estimation. In order to derive an upper bound on the possible accuracy of DTMs, the test set was used to guide the search, and the termination condition for the best-first search was changed so that a maximum of 30 consecutive nodes need not show improvement before we stop (up from five).

Because the search is still a heuristic search, the best feature subset might not be found, and so this is not a true upper bound (it is a lower bound on the upper bound). As shown in the next section, the heuristic sometimes fails to find the best node. The upper bound usually leads to optimistic accuracy estimates, especially if the test set size is small and there are many features that allow perfectly fitting the test set (*e.g.*, the lung-cancer dataset). The upper bound does not show that the given accuracy is achievable, something we cannot expect, but rather that performance above this level is impossible without changing the hypothesis space or improving the search for the best feature subset.

[3] Although IDTM performs cross-validations internally, the outer cross-validation is completely independent.

Dataset	Features	Train sizes	Test	Majority Accuracy	C4.5 Accuracy	IDTM Accuracy	IDTM* Accuracy
audiology	69	226	CV	25.2±2.8	**79.3±3.5**	71.3± 3.8	88.0±1.9
breast-cancer	9	286	CV	70.4±2.3	73.9±2.8	75.3± 2.4	83.7±1.9
chess	36	3196	CV	52.2±1.1	**99.5±0.1**	97.8± 0.2	98.4±0.2
corral	6	32	128	56.3±4.4	81.2±3.5	**100.0±0.0**	100.0±0.0
dna	180	2000	1186	50.8±1.5	92.3±0.8	**94.6±0.7**	94.9±0.6
lenses	4	24	CV	65.0±8.4	83.3±7.0	83.3± 7.0	91.7±5.7
lung-cancer	56	32	CV	41.7±9.0	49.2±7.5	53.3± 9.9	100.0±0.0
Monk1	6	124	432	50.0±2.4	75.7±2.1	**100.0±0.0**	100.0±0.0
Monk2	6	169	432	67.1±2.3	65.0±2.3	64.4± 2.3	81.9±1.9
Monk2-local	17	169	432	67.1±2.3	70.4±2.2	**100.0±0.0**	67.1±2.3
Monk3	6	122	432	47.2±2.4	97.2±0.8	97.2± 0.8	97.2±0.8
mushroom	22	8124	CV	51.8±0.5	100.0±0.0	99.9± 0.4	100.0±0.0
parity5+5	10	100	1024	50.0±1.6	50.0±1.6	**100.0±0.0**	50.0±1.6
tic-tac-toe	9	958	CV	65.4±1.7	**85.6±1.1**	78.2± 1.4	84.8±0.9
vote	16	435	CV	61.4±2.1	**95.6±0.5**	94.3± 0.4	99.1±0.4
vote1	15	435	CV	61.4±2.1	88.0±1.8	87.6±1.3	97.0±0.7

Table 1. Comparison of majority, C4.5, IDTM, and IDTM* on discrete domains. Bold indicates significantly better accuracies (either C4.5 or IDTM.)

4.2 Experimental Results

Table 1 shows the accuracy results for the following induction algorithms:

1. A majority induction algorithm, which simply predicts the majority class. The accuracy shown is sometimes referred to as baseline accuracy.
2. The C4.5 decision-tree induction algorithm with the default parameter settings.
3. The IDTM induction algorithm described in this paper.
4. The IDTM* induction "algorithm," which gives an approximate upper bound on the performance of any induction algorithm using DTMs. We stress that this is not really a learning algorithm because it is given access to the test set.

The numbers after the "±" sign indicate the standard deviation of the reported accuracy. On cross-validated runs, the standard deviation of the fold accuracies is given; on runs that had a pre-specified test-set, the standard accuracy is the computed according to the Binomial model which assumes each test set instance is an independent Bernoulli experiment, and thus the standard deviation of the mean accuracy is $\sqrt{\mathrm{acc}(1-\mathrm{acc})/m}$ (see Breiman et al. (1984), Devijver & Kittler (1982)).

The results demonstrate that IDTM can achieve high accuracy in discrete domains using the simple hypothesis space of DTMs. In corral, dna, the Monk

Dataset	Feat-ures	Train sizes	Test	Majority Accuracy	C4.5 Accuracy	IDTM Accuracy	IDTM* Accuracy
australian	14	690	CV	55.5±2.3	85.4±1.1	84.9± 1.7	89.4±1.3
breast	10	699	CV	65.5±1.7	**95.4±0.7**	90.6± 0.9	96.1±0.6
cleve	13	303	CV	54.4±3.6	72.3±2.2	75.5± 3.2	90.8±2.2
crx	15	690	CV	55.5±2.0	85.9±1.4	86.7± 1.1	89.1±1.2
diabetes	8	768	CV	65.1±1.6	**71.8±1.0**	66.0± 1.1	71.0±1.2
german	24	1000	CV	70.0±1.3	69.8±1.1	69.4± 1.1	81.4±1.3
glass	9	214	CV	35.5±3.3	**65.5±3.2**	41.6± 3.0	55.6±1.8
glass2	9	163	CV	53.3±4.0	**70.6±2.0**	48.9± 4.0	69.2±2.8
hayes-roth	4	160	CV	31.8±2.6	**64.8±4.6**	57.5± 3.2	76.9±2.1
heart	13	270	CV	55.6±3.1	76.7±1.8	**80.4± 1.6**	91.5±1.5
hepatitis	19	155	CV	79.2±3.9	80.0±3.7	77.9± 2.8	96.0±2.0
horse-colic	22	368	CV	63.1±2.3	85.1±1.2	84.3± 0.7	92.1±0.9
hypothyroid	25	3163	CV	95.2±0.4	**99.1±0.2**	97.0± 0.4	97.9±0.3
iris	4	150	CV	23.3±2.5	95.3±1.4	94.7± 1.3	94.7±1.3
labor-neg	16	57	CV	65.3±7.7	**85.7±3.5**	75.3± 7.6	98.3±1.7
letter	16	15000	5000	3.7±0.3	**86.8±0.5**	69.2± 0.7	69.2±0.7
lymphography	18	148	CV	54.6±6.4	78.4±1.7	76.2± 3.6	93.9±1.61
satimage	36	4435	2000	23.1±0.9	**85.2±0.8**	78.9±0.9	78.9±0.9
segment	19	2310	CV	11.1±0.4	**96.4±0.3**	56.3±1.3	57.5±1.2
sick-euthyroid	25	3163	CV	90.7±0.6	**97.7±0.3**	94.9± 0.3	96.0±0.3
soybean-small	35	47	CV	36.0±6.7	100.0±0.0	97.5± 2.5	100.0±0
vehicle	18	846	CV	22.6±1.0	**69.8±1.8**	59.3± 1.5	63.4±1.4

Table 2. Comparison of majority, C4.5, IDTM, and IDTM* on non-discrete domains. Bold indicates significantly better accuracies (either C4.5 or IDTM.)

problems, and parity, IDTM significantly outperforms C4.5 (a difference of more than two standard deviations). In audiology, chess, tic-tac-toe, and vote, performance is significantly below that of C4.5. Performance is approximately the same for the rest.

The $94.6\% \pm 0.7\%$ accuracy of IDTM on the DNA dataset, containing 180 binary features, 2,000 training instances, and 1,186, test instances, is higher than many other state-of-the-art induction algorithms reported for this dataset in (Taylor et al. 1994). For example, CART (Breiman et al. 1984) achieves 91.5% accuracy, Backprop (Rumelhart, Hinton & Williams 1986) achieves 91.2% accuracy, CN2 (Clark & Niblett 1989) achieves 90.5% accuracy, and k-nearest neighbor achieves 84.5% accuracy.

Table 2 shows some results for datasets containing continuous features. In these domains, we expected IDTM to fail miserably, given that the chances of matching continuous features in the table are slim without preprocessing the data. Although C4.5 clearly outperforms IDTM on most datasets, IDTM outperforms C4.5 on the heart dataset and achieves similar performance on nine out of the 22 datasets (australian, cleve, crx, german, hepatitis, horse-colic, iris, lymphography, and soybean).

Running times on a Sparc 10 varied from about one minute for the Monk datasets to 15 hours for the dna dataset. The long running time for the dna dataset was due to the branching factor of 180 in the feature-subset space.

4.3 Discussion

We believe that best-first search is doing a reasonable job at searching the space for good feature-subset candidates. The dna dataset is a clear example; out of $2^{180} = 1.5 \cdot 10^{54}$ possible subsets, only 21 nodes were expanded, resulting in a graph with 3,723 nodes, each representing a feature subset. Two interesting examples where best-first search fails to find a good subset are the Monk2-local and the parity5+5 problems. In these datasets, the IDTM* algorithm fails to find a feature subset at least as good as the subset chosen by IDTM. Monk2-local is a local encoding of the original Monk2 problem where out of eighteen features, only six are relevant. The baseline (majority) is 67.13% accuracy. All 268 nodes that were expanded did not result in any improvement, so the algorithm halted. IDTM, on the other hand, climbed a path of seemingly improving nodes, and found the correct subset. Parity5+5 is parity of five features with five random features; a similar event happened in this case.

The fact that IDTM's performance equals that of C4.5 in domains with continuous features indicates that many such features are not very useful, or that they contain few values, or that C4.5 is not using the information contained in them. The soybean dataset contains only one feature with more than four values, even though all are declared continuous. The german dataset contains 21 continuous features that have less than five values each (out of a total of 24 features); IDTM indeed chooses only the features with a few values. Iris contains over 20 to 43 values for each continuous feature, yet a table using a single feature—petal width—has 94.7% accuracy. The crx dataset contains six continuous features, five having more than 130 values each; in our experiments, IDTM usually used three non-continuous features that contained a total of 18 possible values (making the 18 line table extremeley easy to comprehend).

We conjecture that IDTM algorithm outperforms C4.5 in discrete domains when the features interact and not too many features are relevant. Decision trees are well suited for local relevances (*i.e.*, different features are relevant in different regions of the instance space), but the greedy top-down recursive partitioning algorithms tend to fail when features interact. DTMs are suited to concepts where some features are globally relevant; the feature subset selection algorithm used here is conducting a best-first search and is thus able to capture interactions. The tic-tac-toe dataset is an example where features are locally relevant; the Monk1, Monk2, and parity5+5 datasets have feature interactions.

5 Related Work

Because they permit one to display succinctly the conditions that must be satisfied before prescribed actions are to be performed, decision tables are becoming popular in computer programming and system design as devices for organizing logic.
—Reinwald & Soland (1966)

In the early sixties, algorithms were created to convert decision tables into optimal computer programs (decision trees) under different measure of optimality using branch and bound procedures (Reinwald & Soland 1966, Reinwald & Soland 1967). In the early seventies, these procedures were improved using dynamic programming techniques (Garey 1972, Schumacher & Sevcik 1976). Hyafil & Rivest (1976) showed that building an optimal decision tree from instances (or from a table) is NP-complete. Hartmann, Varshney, Mehrotra & Gerberich (1982) show how to convert a decision table into a decision tree using mutual information. The algorithm is very similar to ID3 (Quinlan 1986). All these approaches, however, dealt with conversions that are information preserving, *i.e.*, all entries in the table are correctly classified and the structures are not used for making predictions.

The rough sets community has been using the hypothesis space of decision tables for a few years (Pawlak 1987, Pawlak 1991, Slowinski 1992). Researchers in the field of rough sets suggest using the degrees-of-dependency of a feature on the label (called γ) to determine which features should be included in a decision table (Ziarko 1991, Modrzejewski 1993). Another suggestion was to use normalized entropy (Pawlak, Wong & Ziarko 1988), which is similar to the information gain measure of ID3. These approaches ignore the utility of the specific features to the specific induction algorithm and to the hypothesis space used.

Much work in the rough sets community has focused on finding the **core** set of features, which are the indispensable features, and **reducts** which are sets of features that allow a Bayesian classifier to achieve the highest possible accuracy. The core is the set of **strongly relevant** features as described in John et al. (1994). While finding reducts is appealing in theory, they are not necessarily optimal subsets for a given induction algorithm. Much of the work presented here stemmed from a paper claiming that reducts are not necessarily useful (Kohavi & Frasca 1994).

Almuallim & Dietterich (1991) described the FOCUS algorithm which is equivalent to finding the DTM with the smallest number of features in the schema and with no conflicting instance projections. The main problem with FOCUS algorithm is that it has no way of dealing with noise and thus the table overfits the data. The algorithm is also unable to handle continuous features.

Almuallim & Dietterich (1992) discussed the "Multi-balls" algorithm that has high coverage—for a given sample size, the number of concepts it can PAC learn is close to the upper bound of any learning algorithm. DTMs can be viewed as a multi-balls hypothesis space because the centers are equidistant. However, the induction method is completely different and the maximal number of balls a DTMs creates is less than the upper bound given by the Gilbert-Varshamov bound.

A nearest-neighbor algorithm can be viewed as a generalization of a DTM with zero weights on the features not included in the schema. However, while nearest-neighbor algorithms use the nearest neighbor to classify instances, a DTM classifier defaults to the majority whenever the distance is greater than zero.

Feature subset selection has been long studied in the statistics community (Miller 1990, Boyce, Farhi & Weischedel 1974), in the pattern recognition community (Devijver & Kittler 1982), and lately in the machine learning community (John et al. 1994, Moore & Lee 1994, Caruana & Freitag 1994, Kohavi & Frasca 1994, Langley & Sage 1994, Aha & Bankert 1994).

Decision tables have a bias similar to that of oblivious read-once decision graphs (OODGs) (Kohavi 1994*a*, Kohavi 1994*b*): all of the features chosen for the schema are tested during classification. This implies that it is easy to convert a decision table into an OODG, perhaps making it more comprehensible.

6 Summary and Future Work

In this paper, we showed that a decision table with a default rule mapping to the majority class can be used to classify instances in discrete spaces with accuracy that is sometimes higher than state-of-the-art induction algorithms. Decision tables are easy for humans to understand, especially if not too big (*e.g.*, the decision table for the crx dataset has 18 entries). For future classification, the resulting decision table provides a constant classification time on the average (using a hash table storing the majority) and is therefore well suited for applications in real-time environments.

We observed that on some datasets with continuous values, the prediction accuracy of IDTM is comparable to that of C4.5. This observation indicates that some real-world datasets from the StatLog and the UC Irvine repositories do not have much "information" in the real values or that C4.5 is unable to utilize this information.

Our goal in this paper has not been to claim that decision tables provide a very good hypothesis space for induction algorithms; rather, we have shown that such a simple hypothesis space can lead to high performance, a point previously made by Holte (1993), although he used a different algorithm. The IDTM algorithm described here performs better than Holte's algorithm and sometimes outperforms C4.5.

Generalization without a bias is impossible (Schaffer 1994, Wolpert 1994). IDTM is biased to select a feature subset maximizing cross-validation accuracy estimates. When the estimates are good, IDTM should choose a feature subset that leads to high prediction accuracy. Our empirical evidence indicates that the estimates are usually good, but have high variability.

We also formalized the idea of an **incremental cross-validation** algorithm, which is applicable whenever an induction algorithm supports incremental add and delete operations. We used this approach to show how cross-validation of an IDTM and a dataset can be performed in time that is linear in the number of instances, the number of features, and the number of label values. We suggested that incremental cross-validation can lead to fast accuracy estimates for decision-tree induction algorithm.

IDTM may be used to select a subset of features that yield good performance and which can provide a starting point for a feature subset selection search that uses a more complex algorithm. It is also possible to test constructive induction methods that construct new features (*e.g.*, interval discretization) with this method.

Acknowledgments The work in this paper was done using the $\mathcal{MLC}$++ library, partly funded by ONR grant N00014-94-1-0448 and NSF grants IRI-9116399 and IRI-9411306. Special thanks to Brian Frasca and Yeogirl Yun who implemented some of the classes in $\mathcal{MLC}$++ that made this paper possible. Rob Holte provided excellent comments and useful pointers after seeing an earlier version of this paper. Scott Benson, Tom Dietterich, Jerry Friedman, George John, Pat Langley, Ofer Matan, Ross Quinlan, Rob Tibshirani, Paul Utgoff, and one anonymous referee provided useful comments. George Light and Pat Langley helped clear the presentation style.

References

Aha, D. W. & Bankert, R. L. (1994), A comparative evaluation of sequential feature selection algorithms, *in* "Proceedings of the Fifth International Workshop on Artificial Intelligence and Statistics", pp. 1–7.

Almuallim, H. & Dietterich, T. G. (1991), Learning with many irrelevant features, *in* "Ninth National Conference on Artificial Intelligence", MIT Press, pp. 547–552.

Almuallim, H. & Dietterich, T. G. (1992), On learning more concepts, *in* "Proceedings of the Ninth International Conference on Machine Learning", Morgan Kaufmann, pp. 11–19.

Boyce, D., Farhi, A. & Weischedel, R. (1974), *Optimal Subset Selection*, Springer-Verlag.

Breiman, L., Friedman, J. H., Olshen, R. A. & Stone, C. J. (1984), *Classification and Regression Trees*, Wadsworth International Group.

Caruana, R. & Freitag, D. (1994), Greedy attribute selection, *in* W. W. Cohen & H. Hirsh, eds, "Machine Learning: Proceedings of the Eleventh International Conference", Morgan Kaufmann.

Clark, P. & Niblett, T. (1989), "The CN2 induction algorithm", *Machine Learning* **3**(4), 261–283.

Cormen, T. H., Leiserson, C. E. & Rivest, R. L. (1990), *Introduction to algorithms*, McGraw-Hill.

Devijver, P. A. & Kittler, J. (1982), *Pattern Recognition: A Statistical Approach*, Prentice-Hall International.

Efron, B. (1983), "Estimating the error rate of a prediction rule: improvement on cross-validation", *Journal of the American Statistical Association* **78**(382), 316–330.

Garey, M. R. (1972), "Optimal binary identification procedures", *Siam Journal on Applied Mathematics* **23**, 173–186.

Ginsberg, M. L. (1993), *Essential of Artificial Intelligence*, Morgan Kaufmann.

Hartmann, C. R. P., Varshney, P. K., Mehrotra, K. G. & Gerberich, C. L. (1982), "Application of information theory to the construction of efficient decision trees", *IEEE Transactions on information theory* **IT-28**(4), 565–577.

Holte, R. C. (1993), "Very simple classification rules perform well on most commonly used datasets", *Machine Learning* **11**, 63–90.

Hyafil, L. & Rivest, R. L. (1976), "Constructing optimal binary decision trees is NP-complete", *Information Processing Letters* **5**(1), 15–17.

John, G. H. (1994), Cross-validated C4.5: Using error estimation for automatic parameter selection, Technical Report STAN-CS-TN-94-12, Computer Science Department, Stanford University.

John, G., Kohavi, R. & Pfleger, K. (1994), Irrelevant features and the subset selection problem, *in* "Machine Learning: Proceedings of the Eleventh International Conference", Morgan Kaufmann, pp. 121–129. Available by anonymous ftp from: `starry.Stanford.EDU:pub/ronnyk/ml94.ps`.

Kohavi, R. (1994*a*), Bottom-up induction of oblivious, read-once decision graphs, *in* "Proceedings of the European Conference on Machine Learning". Available by anonymous ftp from `starry.Stanford.EDU:pub/ronnyk/euroML94.ps`.

Kohavi, R. (1994*b*), Bottom-up induction of oblivious, read-once decision graphs : strengths and limitations, *in* "Twelfth National Conference on Artificial Intelligence", pp. 613–618. Available by anonymous ftp from `Starry.Stanford.EDU:pub/ronnyk/aaai94.ps`.

Kohavi, R. (1994*c*), Feature subset selection as search with probabilistic estimates, *in* "AAAI Fall Symposium on Relevance", pp. 122–126. Available by anonymous ftp from: `starry.Stanford.EDU:pub/ronnyk/aaaiSymposium94.ps`.

Kohavi, R. & Frasca, B. (1994), Useful feature subsets and rough set reducts, *in* "Third International Workshop on Rough Sets and Soft Computing", pp. 310–317. Available by anonymous ftp from: `starry.Stanford.EDU:pub/ronnyk/rough.ps`.

Langley, P. & Sage, S. (1994), Induction of selective bayesian classifiers, *in* "Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence", Morgan Kaufmann, Seattle, WA, pp. 399–406.

Maron, O. & Moore, A. W. (1994), Hoeffding races: Accelerating model selection search for classification and function approximation, *in* "Advances in Neural Information Processing Systems", Vol. 6, Morgan Kaufmann.

Miller, A. J. (1990), *Subset Selection in Regression*, Chapman and Hall.

Modrzejewski, M. (1993), Feature selection using rough sets theory, *in* P. B. Brazdil, ed., "Proceedings of the European Conference on Machine Learning", pp. 213–226.

Moore, A. W. & Lee, M. S. (1994), Efficient algorithms for minimizing cross validation error, *in* W. W. Cohen & H. Hirsh, eds, "Machine Learning: Proceedings of the Eleventh International Conference", Morgan Kaufmann.

Moore, A. W., Hill, D. J. & Johnson, M. P. (1992), An empirical investigation of brute force to choose features, smoothers and function approximators, *in* "Computational Learning Theory and Natural Learning Systems Conference".

Murphy, P. M. & Aha, D. W. (1994), UCI repository of machine learning databases, For information contact ml-repository@ics.uci.edu.

Nilsson, N. J. (1980), *Principles of Artificial Intelligence*, Morgan Kaufmann.

Pawlak, Z. (1987), "Decision tables — a rough sets approach", *Bull. of EATCS* **33**, 85–96.

Pawlak, Z. (1991), *Rough Sets*, Kluwer Academic Publishers.

Pawlak, Z., Wong, S. & Ziarko, W. (1988), "Rough sets: Probabilistic versus deterministic approach", *Internation Journal of Man Machine Studies* **29**, 81–95.

Quinlan, J. R. (1986), "Induction of decision trees", *Machine Learning* **1**, 81–106. Reprinted in Shavlik and Dietterich (eds.) Readings in Machine Learning.

Quinlan, J. R. (1993), *C4.5: Programs for Machine Learning*, Morgan Kaufmann, Los Altos, California.

Reinwald, L. T. & Soland, R. M. (1966), "Conversion of limited-entry decision tables to optimal computer programs i: Minimum average processing time", *Journal of the ACM* **13**(3), 339–358.

Reinwald, L. T. & Soland, R. M. (1967), "Conversion of limited-entry decision tables to optimal computer programs ii: Minimum storage requirement", *Journal of the ACM* **14**(4), 742–755.

Rumelhart, D. E., Hinton, G. E. & Williams, R. J. (1986), *Learning Internal Representations by Error Propagation*, MIT Press, chapter 8.

Schaffer, C. (1994), A conservation law for generalization performance, *in* "Machine Learning: Proceedings of the Eleventh International Conference", Morgan Kaufmann, pp. 259–265.

Schumacher, H. & Sevcik, K. C. (1976), "The synthetic approach to decision table conversion", *Communications of the ACM* **19**(6), 343–351.

Shao, J. (1993), "Linear model seletion via cross-validation", *Journal of the American Statistical Association* **88**(422), 486–494.

Slowinski, R. (1992), *Intelligent decision support : handbook of applications and advances of the rough sets theory*, Kluwer Academic Publishers.

Stone, M. (1974), "Cross-validatory choice and assessment of statistical predictions", *Journal of the Royal Statistical Society B* **36**, 111–147.

Taylor, C., Michie, D. & Spiegalhalter, D. (1994), *Machine Learning, Neural and Statistical Classification*, Paramount Publishing International.

Thrun etal. (1991), The monk's problems: A performance comparison of different learning algorithms, Technical Report CMU-CS-91-197, Carnegie Mellon University.

Utgoff, P. E. (1994), An improved algorithm for incremental induction of decision trees, *in* "Machine Learning: Proceedings of the Eleventh International Conference", Morgan Kaufmann, pp. 318–325.

Weiss, S. M. (1991), "Small sample error rate estimation for k-nearest neighbor classifiers", *IEEE Transactions on Pattern Analysis and Machine Intelligence* **13**(3), 285–289.

Weiss, S. M. & Kulikowski, C. A. (1991), *Computer Systems that Learn*, Morgan Kaufmann, San Mateo, CA.

Wolpert, D. H. (1994), The relationship between PAC, the statistical physics framework, the Bayesian framework, and the VC framework, Technical report, The Santa Fe Institute, Santa Fe, NM.

Zhang, P. (1992), "On the distributional properties of model selection criteria", *Journal of the American Statistical Association* **87**(419), 732–737.

Ziarko, W. (1991), The discovery, analysis, and representation of data dependencies in databases, *in* G. Piatetsky-Shapiro & W. Frawley, eds, "Knowledge Discovery in Databases", MIT Press.