An Intelligent Assistant for the Knowledge Discovery Process*

Abraham Bernstein and Foster Provost
{abernste|fprovost}@stern.nyu.edu

Abstract

A knowledge discovery (KD) process involves preprocessing data, choosing a data-mining algorithm, and postprocessing the mining results. There are very many choices for each of these stages, and non-trivial interactions between them. Consequently, both novices and data-mining specialists need assistance in navigating the space of possible KD processes. We present the concept of Intelligent Discovery Assistants (IDAs), which provide users with (i) systematic enumerations of valid KD processes, so important, potentially fruitful options are not overlooked, and (ii) effective rankings of these valid processes by different criteria, to facilitate the choice of KD processes to execute. We use a prototype to show that an IDA can indeed provide useful enumerations and effective rankings.

1 Introduction[1]

The Knowledge Discovery (KD) process is one of the central notions of the field of Knowledge Discovery and Data mining. For this paper, we will consider three KD process stages: preprocessing data, the application of an induction algorithm, and the post-processing of the output. Figure 1 shows three example KD processes.[2] Process 1 comprises simply the application of a decision-tree inducer. Process 2 preprocesses the data by applying discretization of numeric attributes, and then builds a naïve Bayesian classifier. Process 3 preprocesses the data first by taking a random subsample, then applies discretization, and then builds a naïve Bayesian classifier.

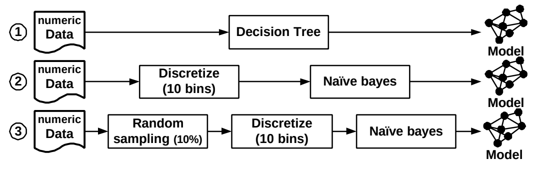

Figure 1. Three valid KD processes

We introduce the notion of Intelligent Discovery Assistants (IDAs), which help data miners with the exploration of the space of *valid* KD processes. A valid KD process violates no fundamental constraints of its constituent techniques. For example, if the input data set contains numeric attributes, simply applying naïve Bayes is not a valid KD process (because naïve Bayes applies only to categorical attributes). However, Process 2 is valid, because it preprocesses the data with a discretization routine, transforming the numeric attributes to categorical ones.

IDAs rely on an explicit ontology of knowledge discovery techniques. For our purposes, the KD ontology defines the existing techniques and their properties. Given such an ontology, an IDA can perform a search of the space of valid processes, considering techniques to be operators that change the world state, with preconditions that constrain their applicability. Figure 2 shows some (simplified) ontology entries (cf., Figure 1).

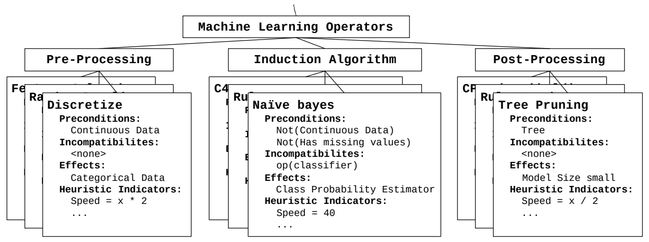

Figure 2. Simplified elements of a KD ontology

Above we said that an IDA *helps* a data miner. More specifically, an IDA determines characteristics of the data and of the desired mining result, and enumerates the KD processes that are valid for producing the desired result from the given data. Then the IDA provides the user assistance in choosing processes to execute, by ranking the process (heuristically) according to what is important to the user.

Results should be ranked differently for different users. The ranking in Figure 1 is based on the number of techniques that form the plan. If the user were interested in minimizing fuss, this ranking would be a useful suggestion. A different user may want to minimize run time, e.g., to get some results quickly. In that case, the reverse of the ranking shown in Figure 1 would be appropriate. There are many other ranking criteria: accuracy, cost sensitivity, comprehensibility, etc., or some combination thereof.

In this paper, we claim that IDAs can provide users with three benefits:

1. a systematic enumeration of valid KD processes, so they do not miss important, potentially fruitful options;
2. effective rankings of these valid processes by different criteria, to help them choose between the options;
3. an infrastructure for sharing KD knowledge, which leads to what economists call network externalities.

We support the first claim by presenting in more detail the design of effective IDAs, including a working prototype. We then provide further support by showing some of the plans that the prototype produces, and arguing that they

[2] Descriptions of all of the techniques can be found in a data-mining textbook [Witten & Frank, 2000].

* Working Paper of the Center for Digital Economy Research, New York University – Leonard Stern School of Business, CeDER Working Paper # IS-01-01
also to be presented at the IJCAI 2001 Workshop on Wrappers for Performance Enhancement in Knowledge Discovery in Databases

would be useful even to expert data miners. We provide support for the second claim with an empirical study with some example ranking heuristics. We show that we can rank quite well (prospectively) by speed, and sometimes well even by accuracy. We provide support for the third claim with a simple example of how an IDA would be especially helpful to a team of data miners.

2 Intelligent Discovery Assistants

Although domain-specific elements could be incorporated into IDAs, for clarity and generality we concentrate on domain-independent elements of the KD process. For example, when presented with a data set to mine, a knowledge-discovery worker (researcher or practitioner) is faced with a confusing array of choices [Witten & Frank, 2000]: should I use C4.5 or naive Bayes or logistic regression? Should I use discretization? If so, what method? Should I subsample? Should I prune? How do I take into account costs of misclassification?

2.1 Ontology-based IDAs

The overall process followed by an IDA is shown in Figure 3. An IDA interacts with the user to obtain data, goals and desiderata. Then it composes the set of valid KD processes, according to the constraints implied by the user inputs, the data, and/or the ontology. This involves choosing induction algorithm(s), and appropriate pre- and post-processing modules (as well as other aspects of the process, not considered in this paper). Next, the IDA will rank the suitable processes into a suggested order based on the user's desiderata. The user can select plans from the suggestion list. Finally, the IDA will produce code for and can execute (automatically) the suggested processes.

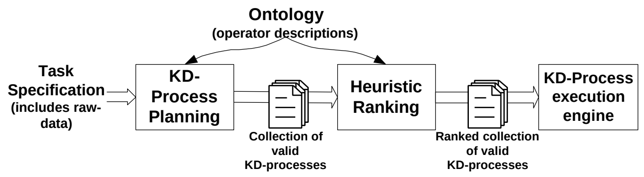
Ontology
(operator descriptions)
Task Specification (includes raw-data)
KD-Process Planning
Collection of valid KD-processes
Heuristic Ranking
Ranked collection of valid KD-processes
KD-Process execution engine

Figure 3: The overall process followed by an IDA

Consider a straightforward example. A user presents a very large data set, including both numeric and categorical data, and specifies *classification* as the learning task (along with the appropriate dependent variable). The IDA asks the user to specify desired tradeoffs between accuracy and speed of learning (these are just a couple of possibilities). Then the IDA determines, of all the possible KD processes, which are appropriate. With a small ontology, there might be few, with a large ontology there might be many. For this task, decision-tree learning alone might be appropriate. Or, a decision-tree program plus subsampling as a preprocess, or plus pruning as a post-process, or plus both. Are naive Bayes or logistic regression appropriate for this example? Not by themselves. Naive Bayes only takes categorical data. Logistic regression takes only numeric. However, a KD process with appropriate preprocessing may include them (transforming the data type), and may do better than the decision tree. What if the user is willing to trade some accuracy to get results fast?

The IDA uses the ontology to assist the user in composing valid and useful KD processes. In our prototype, for each operator the ontology contains:

- Human-readable information about each of the operators.
- A specification of the conditions under which the operator can be applied. This contains both a pre-condition on the state of the KD process as well as its compatibility with preceding operators.
- A specification of the operator's effects on the KD process's state and its data.
- Estimations of the operator's influence on attributes like speed, accuracy, model comprehensibility, etc.

In addition, the ontology groups the operators into logical groups, which can be used to narrow the set of operators to be considered at each stage of the KD process. Figure 4 shows a structural view of our prototype ontology. It groups the KD operators into three major groups: preprocessing, induction, post-processing. Each of these groups is further sub-divided. At the leaves of this tree are the actual operators (not shown in the figure, except for one example: C4.5). For example, the induction algorithm group is subdivided into classifiers, class probability estimators (CPE), and regressors. Classifiers are further grouped into decision trees and rule learners, and the first of those groups includes C4.5 [Quinlan, 1993].

2.2 IDEA: A Prototype IDA

The **I**ntelligent **D**iscovery **E**lectronic **A**ssistant (IDEA) is a prototype IDA that uses our ontology-based approach. Following our general framework for IDAs (see Figure 3) IDEA gathers a *task specification* for the KD process. First IDEA analyzes the data that the user wishes to mine and extracts the relevant meta-information about the data, such as the types of attributes included (e.g., continuous, categorical). Using a GUI, the user then can complement the

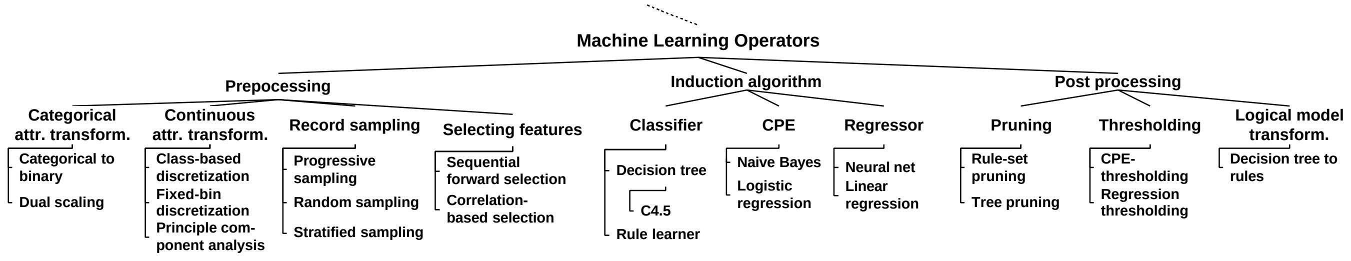

Figure 4: The Data-mining Ontology (partial view)

gathered information with additional knowledge about the data (such as structural attributes IDEA could not derive from the metadata), and can specify the type of information/model he/she wishes to mine and desired tradeoffs (speed, accuracy, cost sensitivity, comprehensibility, etc.).

IDEA's first core component, the *KD-process planner*, then searches for KD processes that are valid given the task specification from within the design space of overall possible KD processes, as defined by the ontology. IDEA searches the space of valid KD processes starting from the start state, finding compatible operators using the compatibilities (the pre-conditions), and transforming the state using the operators' effects. Given the limited number of operators in the prototype KD-ontology (currently a few dozen) the KD-process planner currently can generate all valid processes (up to several hundred for problems with few constraints) in less than a second.

A collection of valid KD processes typically will contain a series of processes that are undesirable for certain user goals--they make undesirable trade-offs, such as sacrificing accuracy completely to obtain the model fast, etc. IDEA's second core component, the *heuristic ranker*, ranks the valid KD processes using a heuristic function and the user's trade-off preferences as defined by weights entered through the GUI. It computes the rankings of the plans by using the operators' characteristics (such as their influence on speed, accuracy, and model comprehensibility) that are specified in the ontology.

IDEA's GUI allows the user to sort the list of plans using any of the rankings, including the combined ranking derived from applying weights on the different characteristics, and to examine the details of any plan.

3 IDAs can produce useful KD processes

Our first claim is that ontology-based IDAs can enumerate KD processes useful to a data miner. We support our claim in two ways. First, the preceding section described how the ontology can enable the composition of only valid plans (only valid plans will be useful). Second, in this section we describe process instances produced by IDEA, in order to provide evidence that they can be non-trivial. Even with the prototype ontology, IDEA produces good KD process enumerations (sometimes surprisingly good).

3.1 Enumeration using the full ontology

Given the ontology whose structure is shown in Figure 4, the goal of *classification*, and the constraints imposed by the pre- and post-conditions of the operators,[3] IDEA produces 597 valid process instances. It should be clear that for a novice data miner, this process space is overwhelming.[4] Many novice users simply use the algorithm that they are familiar with, with little pre- or post-processing.

[3]These are not shown, but are straightforward constraints such as: logistic regression requires numeric attributes, decision-tree pruning can only apply to decision trees, etc.

[4]Without the operator-specific constraints, using only the ordering imposed by the ontology, the total number of possible process instances is 163,840.

However, although the benefits of IDAs are clearest for novice users, they are valuable even to experts. Consider the 2000 KDDCUP, in which 30 teams of data-mining researchers and professionals competed to mine knowledge from electronic-commerce data [Brodley & Kohavi, 2000]. Most algorithm types were tried by only a small fraction of participants. The only algorithm that was tried by more than 20% of the participants was decision-tree induction.

Consider a specific example. When we give IDEA the goal of producing a *cost-sensitive classifier for a two-class problem*, it produces 146 plans, including using class-probability estimators or regressors and adjusting the output threshold, as well as using class-stratified sampling with any classification algorithm. This is a non-trivial enumeration of KD processes: there are many instances of published research on cost-sensitive learning that do not consider these options.

When we give IDEA the goal of producing *comprehensible classifiers*, the top-ranked plan is: `subsample the instances, do feature selection, use a rule learner, prune the resultant rule set`. Although comprehensibility is a goal of much machine-learning research, we are not aware of this plan being used/suggested. As another plan highly ranked by comprehensibility, which also had a high accuracy ranking, IDEA suggested `run C4.5, convert tree to rules, prune rule set`. This also is a non-trivial plan: it is the process shown by Quinlan [1987] to be good for comprehensibility and high accuracy. Although the addition to the ontology of `convert tree to rules` certainly was influenced by Quinlan's work, we did not "program" the system to produce this plan. IDEA composes plans only based on knowledge of *individual* operators.

	steps	*heuristic rank*	
		accuracy	speed
Plan # 1	c4.5	6	13
Plan # 2	part	1	16
Plan # 3	rs, c4.5	14	5
Plan # 4	rs, part	9	10
Plan # 5	fbd, c4.5	8	11
Plan # 6	fbd, part	5	14
Plan # 7	cbd, c4.5	7	12
Plan # 8	cbd, part	3	15
Plan # 9	rs, fbd, c4.5	16	3
Plan # 10	rs, fbd, part	12	8
Plan # 11	rs, cbd, c4.5	15	4
Plan # 12	rs, cbd, part	10	9
Plan # 13	fbd, nb, cpe	4	6
Plan # 14	cbd, nb, cpe	2	7
Plan # 15	rs, fbd, nb, cpe	13	1
Plan # 16	rs, cbd, nb, cpe	11	2

Legend for operators used in plans	
acronym	*name/algorithm*
rs	Random sampling (result instances = 10% of input inst.)
fbd	Fixed bin discretization (10 bins)
cbd	Class based discretization (Fayyad & Irani's MDL method)
c4.5	C4.5 (using Frank & Wittens J48 implementation)
part	Rule Learner (PART, Frank & Witten)
nb	Naïve Byes (John & Langley)
cpe	CPE-Thresholding post-processor

Table 1: 16 process plans and three rankings

3.2 Enumeration using a restricted ontology

For the experiments in the next section, we restricted the ontology to a subset for which it is feasible to study an entire enumeration of plans. The ontology subset uses seven common pre-processing, post-processing, and induction techniques (for which there were appropriate functions in Weka, see below). The experimental task is to build a *classifier*, and has as its start state a data set *containing at least one numeric attribute*. Even this small ontology produces an interesting variety of KD-process plans. Table 1 shows the list of 16 valid plans for this problem on the left; on the right is a legend describing the

7 operators used. For example, the ontology specifies that naïve Bayes only considers categorical attributes, so the planner had to include a preprocessor that transformed the data. Indeed, although the ontology for the experiments is very small, the diversity of plans is greater than in many research papers.

4 IDAs can produce effective rankings

In order to provide empirical support for our claim that IDAs can produce effective rankings, we implemented a code generator for IDEA that exports any collection of plans, which then can be run (automatically). Currently it generates code for the Weka data-mining toolkit [Witten and Frank, 2000], which also allows us to generate Java code for executing the plans, as well as code for evaluating the resulting models based on accuracy and speed of learning.[5]

Note that in this section we use the empirical rankings to support the claim that IDAs can produce effective rankings. We do not claim that we have studied the production of ranking functions to any depth, and we are sure that better ranking functions exist (however, they can not be used/ studied effectively without an implemented IDA).

The plans in Table 1 are ranked by the number of steps in the plan. This is one option in IDEA, because users who will be executing plans manually may be interested in minimizing fuss. Not surprisingly, as with the KDDCUP, decision-tree learning is at the top of the list.

Table 1 also shows the heuristic ranks for both accuracy and speed for our test problem. The heuristic ranks were generated by a functional composition based on the ontology. For example, the ontology specifies a base accuracy and speed for each learner, and specifies that random sampling and discretization will reduce accuracy and will increase speed. The heuristic functions are subjective, based on our experience with the different operators and our reading of the literature (e.g., [Lim *et al.*, 2000]). The ranking functions were decided on before we tried Weka, with one exception. Namely, speed ratings differ markedly by implementation, so we ran Weka on one data set (credit-g) to instantiate the base speed for the three learning algorithms and made minor changes to the speed improvement factors for sampling and for discretization. We chose speed and accuracy ranking for the experiments, because they were the two criteria we could evaluate objectively.

Our first experiment examines how well these heuristic rankings perform. We ran the 16 plans on 15 UCI data sets, using ten-fold cross-validation to compute average classification accuracy and average speed. Table 2 summarizes the results of these runs in columns "speed-heuristic" and "accuracy-heuristic" (for ease of comparison, Table 2 also includes results discussed below). Each result column has sub-columns "avg top," "avg bottom," and "t-test." Respectively, these show the average score (speed or accuracy) for the top-8 plans and for the bottom-8 plans, and the result of running a t-test to determine if the difference in the means is significant. These results show clearly that the speed ranking is effective (at this gross level). The accuracy results are less impressive. In some cases, the heuristic ordering works well, in other cases it fails completely. The latter result is not surprising: researchers have been trying for years to predict which algorithm will be more accurate, with little success.

Dataset	**Speed-heuristic**			**Speed-credit-g**		
	avg. top	avg. bott.	T-test	avg. top	avg. bott.	T-test
balance-scale	7.375	61.125	0.01766	6.75	61.75	0.01541
breast-w	11.625	51.5	0.00543	8.875	54.25	0.00173
heart-h	6.625	48.25	0.00934	6	48.875	0.00754
credit-a	13	109.75	0.01800	9.5	113.25	0.01193
heart-c	8.25	50.75	0.01247	8.125	50.875	0.01199
diabetes	12.25	87.75	0.00271	8.125	91.875	0.00084
vehicle	33.625	344.125	0.01203	24.625	353.125	0.00826
colic	10.875	81.75	0.00607	10.875	81.75	0.00607
ionosphere	29.375	173.25	0.01881	13	189.625	0.00523
vowel	59	919.375	0.07017	30.875	947.5	0.05796
sonar	28.125	145.125	0.01377	19	154.25	0.00508
anneal	25.75	182.625	0.00320	20.875	187.5	0.00171
credit-g	24.125	321.125	0.01919	21.375	323.875	0.01736
sick	81.75	804.25	0.00293	61.875	824.125	0.00166
segment	146.25	792.875	0.01227	79.5	859.625	0.00333

Dataset	**Accuracy-heuristic**			**Accuracy-credit-g**		
	avg. top	avg. bott.	T-test	avg. top	avg. bott.	T-test
balance-scale	76.8792	72.7976	0.16063	73.4329	76.2439	0.24948
breast-w	95.4943	95.7143	0.35410	96.088	95.1206	0.04332
heart-h	80.4842	84.5833	0.04266	85.5374	79.5302	0.00269
credit-a	83.7862	86.7857	0.00780	85.6056	84.9664	0.31685
heart-c	79.2984	77.0833	0.27162	81.2917	75.0901	0.03698
diabetes	75.3247	67.2991	0.00050	73.4274	69.1964	0.06126
vehicle	67.1871	48.6285	9.7E-06	56.365	59.4506	0.29640
colic	82.9805	74.0625	3.3E-05	77.3179	79.725	0.19796
ionosphere	89.2014	76.1458	0.05156	84.8353	80.5119	0.29237
vowel	74.7727	39.6667	8.3E-09	56.3295	58.1098	0.42958
sonar	74.8065	68.125	0.21853	72.8423	70.0893	0.37329
anneal	97.7567	85.9549	4.8E-08	91.1026	92.6089	0.32835
credit-g	71.475	70.125	0.25019	73.9625	67.6375	2.2E-05
sick	97.919	95.481	0.00477	96.1073	97.2927	0.11205
segment	93.934	91.1028	0.06534	90.7837	94.2531	0.02760

Table 2: Ranking Heuristics Assessment

Dataset	Speed			Accuracy	
	heur	sampling	credit-g	heur	rs-pred.
balance-scale	0.80797	0.58765	0.86379	0.49729	0.72095
breast-w	0.76844	0.75206	0.80259	0.01340	0.71024
heart-h	0.87856	0.73391	0.90132	-0.29831	0.52024
credit-a	0.81771	0.73212	0.90950	-0.45250	0.23071
heart-c	0.75147	0.63074	0.81476	0.12749	0.46714
diabetes	0.80529	0.61597	0.88362	0.70154	0.14810
vehicle	0.95294	0.71438	0.97941	0.77967	0.19048
colic	0.88435	0.60615	0.92091	0.76677	0.59738
ionosphere	0.76591	0.76135	0.87279	0.61421	0.80643
vowel	0.94253	0.61962	0.94532	0.72536	-0.33048
sonar	0.72435	0.72029	0.81618	0.25629	0.88143
anneal	0.94085	0.70768	0.99018	0.67391	0.41548
credit-g	0.95947	0.70897	0.99726	0.06028	-0.16595
sick	0.90900	0.76503	0.96056	0.79342	0.40476
segment	0.93529	0.61524	0.94118	0.34546	0.85714

Table 3: Spearman Ranks for Heuristic Ranking

In Table 3 the "heur" column presents the correlation between the rankings produced by the heuristics and the actual speed/accuracy, using Spearman's r_s (recall that a perfect rank correlation is 1, no correlation is 0, and a perfectly inverted ranking is -1; we broke ties in the actual values by flipping a coin, and these results are the averages over 100 coin tosses).

These results show that in about half of the domains, the heuristic accuracy rankings are pretty good, but in some domains the heuristic accuracy rankings are worse than random! However, it is important to note that the IDA in fact *can* produce a "perfect" ranking even for accuracy: it can run all the plans, determine the accuracy, and provide

[5]The last operator in Table 1, `cpe`, which places an appropriate threshold on a class-probability estimator, becomes a no-op in the Weka implementation, because `nb` thresholds automatically.

the user with the ranking. This of course would be time consuming for large data sets or long lists of plans, so an important question is: what is the user's desired tradeoff between accuracy and time? We return to this below.

For our purpose, the speed results are more interesting, because it does not make sense to run all the plans to determine which to suggest. The rank correlations are impressive, but the results deserve closer inspection. Examining the plans we find that six of the eight top-ranked plans start by randomly sampling a subset of 10% of the training data. One would expect these to be significantly faster than the non-sampled plans (interestingly, the two `rs` plans that use `PART` are predicted to be in the slower half of the speed ranking). Therefore, it is important to ask whether the success of the heuristic ranking is due simply to the presence of sampling in the faster plans.

The "sampling" column in Table 3 shows the r_s values if the predicted ranking is based solely on placing all plans using `rs` at the top of the speed ranking and the non-`rs` plans at the bottom (otherwise ranked randomly). These results are striking: the heuristic ranking is in every case better, and in some cases much better than can be explained solely by the inclusion of random sampling. Furthermore, for some domains the heuristic ranking is close to perfect. We also include one additional experiment: an alternative ranking heuristic is simply to use the speed ranking from one domain to predict the rankings for the others. Since we examined credit-g to initialize our heuristic, why not just use the credit-g ranking? In fact, as Table 3 shows (in column "credit-g"), this produces a better ranking than our heuristic does (although it may not be practicable for very large ontologies). In fact, in anneal it is very nearly perfect.

We conclude from these experiments that IDAs can in fact produce effective rankings, although how to do so well certainly deserves further study (see next section).

5 On-going work: sampling for accuracy

Our long-term goal is not simply to be able to rank by speed or by accuracy, but to allow users to specify desired tradeoffs between different criteria. For example, what if I'm willing to trade off a little speed for a better accuracy ranking (but don't want to run *all* the plans). Why not let the IDEA run some very fast plans for the purpose of inferring the accuracy of the time-consuming ones? (Since it can predict speed pretty well.)

We ran one additional experiment to test whether a rudimentary version of this strategy is effective. In particular, for each plan there is a version that begins with `rs` (random sampling of 10% of the data). For each domain we evaluate the following IDEA strategy: run the `rs` plans, and use the observed ranking for the non-`rs` plans.

In some cases (see Table 3, column “rs-pred.”), the sampling-based ranking works surprisingly well. In others it does not work as well as the heuristic ranking. These results are promising enough (given only a 10% random sample) to consider looking further at sampling as an important component for creating accuracy-ranking heuristics.

There are four domains where neither accuracy ranking does well: heart-h, heart-c, credit-a, credit-g. Examining the actual plan runs yields a surprising explanation. The plans without discretization or random sampling are among the lowest-accuracy plans for these domains! This is not predicted by either ranking, but further argues for the need for an IDA that can experiment for the user (automatically) with different plans.

6 Discussion

IDAs are particularly useful because they are systematic in their exploration of the design space of KD processes. Without such a tool users, even experts, are seldom systematic in their search of the KD-process space, and it is unlikely that any user will consider all possible process plans. Therefore, users may overlook important, useful plans.

Up to this point we have discussed, for emphasis, novice users and expert users. However, this is not a true dichotomy. There is a spectrum of expertise along which users reside. For the most novice, any help with KD process planning will be helpful. For the most expert, an IDA could be useful for double-checking his/her thinking, and for automating previously manual tasks. For others along the spectrum, IDAs will have both types of benefits. In addition, the tool may help to educate the non-expert user. For example, when the system produces a highly ranked plan that the user had not considered previously, the user can examine the ontology, and educate himself on some new aspect of the KD process.

A unique benefit of an IDA based on an <u>explicit</u> ontology is the synergy it can provide between teams of users. Consider the following example. George is a member of a large team of data miners, with several on-going projects. While reading the statistics literature he discovers a technique called *dual scaling* [Nishisato, 1994], a preprocessing operator that transforms categorical data into numeric data, in a manner particularly useful for classification. George codes up a new preprocessor (call it `DS`) and uses it in his work. Such discoveries normally are isolated; they do not benefit the team's other projects. However, if George simply adds `DS` into the IDA, and adds the appropriate entry to the ontology, whenever someone else uses the system plans will be generated that use `DS` (when appropriate). This is an example of what economists call network externalities: users get positive value from other people using the "network." In some cases, these plans will be highly ranked (when `DS` is likely to do a good job satisfying the user's criteria). Thus users get the benefit of others' work automatically--no single member of the group has to be an expert in the entire body of data-mining technology.

Finally, we have discussed how users can learn more about data mining by using an IDA. It is also possible for the IDA to learn about data mining, either on its own or through interaction with the users. Since the system can evaluate the plans it produces, it can evaluate and improve its own ranking functions.

7 Related Work

"Automatic bias selection" involves the selection of one of the following, based in part on feedback from the performance of the learner: vocabulary terms, the induction algorithm itself, components of the induction algorithm, parameters to the induction algorithm [desJardins and Gordon, 1995]. However, bias-selection work generally assumes the goal is accuracy maximization (exceptions are Tcheng et al., [1989] who consider accuracy and speed, and Provost and Buchanan [1995], who consider accuracy, speed, and cost sensitivity). Moreover, none of this work composes KD *processes*, nor is it based on an ontology.

There has been work on selecting individual induction algorithms or subcomponents of algorithms based on certain forms of background knowledge. For example, Brodley [1995] chooses subcomponents to form a hybrid decision tree, based on expert knowledge of algorithm applicability. Brazdil et al. [1994] and Gama & Brazdil [1995] use meta-rules drawn from experimental studies, to help predict which algorithms will be better; the rules looked at measurable characteristics of the data (e.g., number of cases, number of attributes, kurtosis).

Engels *et al.* propose to implement a user-guidance module for KD processes ([Engels, 1996], [Engels *et al.*, 1997], [Wirth *et al.*, 1997], and [Verdenius and Engels, 1997]). In particular, the user-guidance module uses a task/method decomposition [Chandrasekaran *et al.*, 1992] to guide the user through a stepwise refinement of a high-level KD process, in order to help the user to construct *the best* plan using a limited model of operations. Finished plans are compiled into scripts for execution. This work is similar to our approach as it provides the user with assistance when constructing KD processes. In contrast, our approach is based on the notion that *it is very difficult to discern the one best plan*, as data-mining results can be unpredictable, and users' desired tradeoffs might not be easily specified (or even known) at the onset of the investigation. An IDA presents the user with many (all) valid plans to choose from and helps him/her to choose among them (using heuristic rankings).

Kerber *et al.* [1998] document the KD process using active links to visually programmed KD processes and to the rationale for major design choices. They collect these descriptions in a repository. This approach facilitates the reuse of KD processes, resulting in a knowledge management system for KD processes. It is complementary to our approach, as it emphasizes the documentation and retrieval of past knowledge, which could be integrated well with our notion of active support as represented by IDAs.

Many of the component methods necessary for building IDAs have been the subject of recent study, especially in the European community. For example, several research projects have studied the use of experimental comparison to select or to rank individual induction algorithms. This work is closely related to our ranking of KD processes (especially since one may put a conceptual box around a KD process and call it an induction algorithm, although this obscures important issues regarding the composition of processes). Brazdil [1998] summarizes some prior methods. More recently, Brazdil and Soares study the ranking of individual induction algorithms, based on (functions of) their performances on previously seen data sets [Brazdil & Soares, 2000; Soares and Brazdil, 2000]. They compare various methods for ranking, which perform comparably. Petrak [2000] presents a convincing analysis of the effectiveness of using subsamples from the data set in question to predict which learning algorithm will yield the lowest error on the entire data set; the technique works remarkably well—although it should be noted that for large data sets often one can achieve maximal accuracy with a surprisingly small subset of the data (cf., progressive sampling [Provost et al., 1999]).

The StatLog project[6] [Michie et al., 1994] has investigated what induction algorithms to use given particular circumstances. The knowledge generated from such projects could be of great use to populate the ontology, as well as to inform the construction of more advanced heuristic functions for ranking KD processes. The METAL project[7] explicitly focuses on IDA-like systems. However, we are not aware of any existing system that uses background knowledge and/or experimentation to compose and rank KD *processes*, although Brazdil argues that it is important to do so [Brazdil, 1998].

Morik proposes to use a case-based repository to store successful chains of pre-processing operators [Morik, 2000]. [8] As pre-processing chains are partial KD processes, the insights gained should complement our work well, and ideally could be integrated with a system such as IDEA.

The only work of which we are aware that uses an explicit ontology within a meta-level machine-learning system is described by Suyama & Yamaguchi [1998]. As far as we can tell, this system uses the ontology to guide the composition, by genetic programming, of fine-grained induction algorithm components (version space, star, entropy, entropy+information-ratio, etc.)

8 Conclusion

Both novices and specialists need assistance in navigating the space of possible KD processes. We have shown that ontology-based IDAs can generate valid, non-trivial, and sometimes surprisingly interesting KD-process instances. Further, we have given empirical evidence that it is possible for IDAs to rank the process instances effectively by various user criteria.

Acknowledgements

Shawndra Hill and Yan Mao assisted with the implementation of the prototype system. We thank IBM for a Faculty Partnership Award.

[6]see http://www.ncc.up.pt/liacc/ML/statlog/index.html
[7]see
http://www.cs.bris.ac.uk/Research/MachineLearning/metal.html
[8] see http://www-ai.cs.uni-dortmund.de/FORSCHUNG/
PROJEKTE/MININGMART/index.eng.html

References

[Brazdil et al., 1994] P. Brazdil, J. Gama, & B. Henery. Characterizing the Applicability of Classification Algorithms Using Meta-Level Learning. In *Proceeedings of the European Conference on Machine Learning (ECML-94),* pp. 83-102.

[Gama & Brazdil, 1995] J. Gama and P. Brazdil: Characterization of Classification Algorithms. 7th Portuguese Conference on Artificial Intelligence, EPIA '95, pp. 189-200.

[Brazdil, 1998] Pavel. B. Brazdil. Data Transformation and Model Selection by Experimentation and Meta-learning. In Proceedings of ECML-98 Workshop on Upgrading Learning to the Meta-Level: Model Selection and Data Transformation, pp. 11-17.

[Brazdil & Soares, 2000] P. Brazdil & C. Soares. A Comparison of Ranking Methods for Classification Algorithm Selection. In *Proceedings 11th European Conference on Machine Learning (ECML-2000)*: 63-74.

[Breiman *et al.*, 1984] Leo Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. Classification and Regression Trees. Belmont, CA: Wadsworth Internatl. Group. 2000

[Brodley, 1995] C. Brodley. Recursive Automatic Bias Selection for Classifier Construction. *Machine Learning* 20(1-2): 63-94.

[Brodley and Kohavi, 2000] Brodley, C. and R. Kohavi. KDD-Cup 2000 Peeling the Onion. Talk at KDD-2000, Boston, August 2000.

[Chandrasekaran et al., 1992] Chandrasekaran, B., Johnson, T. R., and Smith, J. W. 1992. Task-Structure Analysis for Knowledge Modeling. CACM 35, 9, 124-137

[desJardins and Gordon, 1995] desJardin, M., and D. Gordon (1995) Special Issue on Bias Evaluation and Selection. Machine Learning 20, 1/2.

[Engels, 1996] Engels, R. 1996. Planning Tasks for Knowledge Discovery in Databases; Performing Task-Oriented User-Guidance. Proceedings of the International Conference on Knowledge Discovery & Data Mining AAAI-Press, Portland, OR, 170-175

[Engels et al., 1997] Engels, R., Lindner, G., and Studer, R. 1997. A Guided Tour through the Data Mining Jungle. Proceedings of the 3nd International Conference on Knowledge Discovery in Databases. Newport Beach, CA

[Frank & Witten, 1998] Frank, E., and Witten, I. H. Generating Accurate Rule Sets Without Global Optimization. 1998 International Conf. on Machine Learning, 144-151

[Kerber et al., 1998] Kerber, R., Beck, H., Anand, T., and Smart, B. 1998. Active Templates: Comprehensive Support for the Knowledge Discovery Process. Intl. Conf. on Knowledge Discovery and Data Mining, 244-248

[Lim et al., 2000] Lim, T.-S., Loh, W.-Y., and Shih, Y.-S. A Comparison of Prediction Accuracy, Complexity, and Training Time of Thirty-Three Old and New Classification Algorithms. Machine Learning 40 (3), 203-228.

[Michie et al., 1994] D. Michie, D. Spiegelhalter, and C. Taylor (eds.). *Machine Learning, Neural and Statistical Classification*. Ellis Horwood.

[Morik, 2000] Morik, K., *The Representation Race - Preprocessing for Handling Time Phenomena*. Proceedings of the European Conference on Machine Learning 2000 (ECML 2000), Ramon López de Mántaras and Enric Plaza (ed.), Lecture Notes in Artificial Intelligence, Vol. 1810, Springer Verlag Berlin, 2000.

[Nishisato, 1994] Nishisato, S. *Elements of Dual Scaling: An Introduction to Practical Data Analysis.* Hillsdale: Lawrence Erlbaum Associates

[Provost & Buchanan, 1995] F. Provost & B.. Buchanan. Inductive Policy: The Pragmatics of Bias Selection. Machine Learning 20(1-2): 35-61.

[Quinlan, 1987] J. R. Quinlan: Simplifying Decision Trees. *International Journal of Man-Machine Studies* 27(3): 221-234 (1987)

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

[Petrak, 2000]. J. Petrak. Fast Subsampling Performance Estimates for Classification Algorithm Selection. Technical Report TR-2000-07. Austrian Research Institute for Artificial Intelligence.

[Provost et al. 1999] F. Provost, D. Jensen, and T. Oates. Efficient Progressive Sampling. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 23-32.

[Soares & Brazdil, 2000] C. Soares & P. Brazdil. Zoomed Ranking: Selection of Classification Algorithms Based on Relevant Performance Information. In *Proceedings of Principles of Data Mining and Knowledge Discovery, 4th European Conference (PKDD-2000),* 126-135.

[Suyama & Yamaguchi 1998] A. Suyama & T. Yamaguchi. Specifying and Learning Inductive Learning Systems using Ontologies. In *Working Notes from the 1998 AAAI Workshop on the Methodology of Applying Machine Learning: Problem Definition, Task Decomposition and Technique Selection,* pp. 29-36.

[Tcheng, et al., 1989] D. Tcheng, B. Lambert, S. Lu, L. Rendell: Building Robust Learning Systems by Combining Induction and Optimization. IJCAI 1989: 806-812

[Verdenius and Engels, 1997] Verdenius, F., and Engels, R. A Process Model for Developing Inductive Applications. Proceedings of the Seventh Belgian-Dutch Conference on Machine Learning Tilburg, NL, 119-128

[Wirth et al., 1997] Wirth, R., et al. Towards Process-Oriented Tool Support for KDD. Proceedings of the 1st European Symposium on Principles of Data Mining and Knowledge Discovery Trondheim, Norway

[Witten and Frank, 2000] Witten, I., and E. Frank (2000). Data Mininig: Practical machine learning tools and techniques with Java implementations. San Francisco: Morgan Kaufmann Publishers.