Metalearning - A Tutorial

Christophe Giraud-Carrier

December 2008
Introduction

Metalearning
  Theoretical Considerations
  Practical Considerations
  Rice’s Framework

The Practice of Metalearning
  Choosing the content of $A$
  Constructing the Training Metadata
  Choosing $f$
  Computational Cost of $f$ and $S$
  Choosing $p$
  Choosing the form of the output of $S$

Metalearning Systems
  MiningMart
  Data Mining Advisor
  METALA
  Intelligent Discovery Assistant
  Experiment Database

The Road Ahead
Objectives

What I hope to do with this tutorial:

- Define and motivate metalearning
- Describe the main issues involved in metalearning
- Show some examples of metalearning-inspired systems
- Have a good time all the while
Machine learning focuses on accumulating experience about a specific learning task or application (e.g., medical diagnosis, fraud detection, etc.) so as to improve performance on it.

It is:

- What we eat, drink and sleep
- What makes the world go 'round
- What we prescribe to anyone who would have it

And YET...
The Shoemaker’s Children Syndrome

- Everyone is using Machine Learning!
  - Everyone, that is ...
  - Except us!
- Applied machine learning is guided mostly by hunches, anecdotal evidence, and individual experience
- If that is sub-optimal for our “customers,” is it not also sub-optimal for us?
- Shouldn’t we look to the data our applications generate to gain better insight into how to do machine learning?
- If we are not quack doctors, but truly believe in our medicine, then the answer should be a resounding YES!
A Working Definition of Metalearning

- We shall call *metadata* the type of data that may be viewed as being generated through the application of machine learning.
- We shall call *metalearning* the use of machine learning techniques to build models from metadata.
- Hence, metalearning is concerned with accumulating experience on the performance of multiple applications of a learning system.
- Here, we will be particularly interested in the important problem of metalearning for algorithm selection.
Theoretical Considerations

- No Free Lunch (NFL) theorem / Law of Conservation for Generalization Performance (LCG)
  - When taken across all learning tasks, the generalization performance of any learner sums to 0
- Is Metalearning doomed?
NFL Revisited (1/4)

- Consider the space, $\mathcal{F}$, of functions defined over $\mathbb{B}^3 = \{0, 1\}^3$
- Assume that the instances of set $Tr = \{000, 001, \ldots, 101\}$ are observed, and the instances of set $Te = \mathbb{B}^3 - Tr = \{110, 111\}$ constitute the off-training set (OTS) test set

<table>
<thead>
<tr>
<th></th>
<th>Inputs</th>
<th>$f_1$</th>
<th>$f_2$</th>
<th>$f_3$</th>
<th>$f_4$</th>
<th>$f_5$</th>
<th>$f_6$</th>
<th>$f_7$</th>
<th>$f_8$</th>
<th>$f_9$</th>
<th>$f_{10}$</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Training Set</strong></td>
<td>0 0 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>0 0 1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>0 1 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>0 1 1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>1 0 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>1 0 1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>...</td>
</tr>
<tr>
<td><strong>Test Set</strong></td>
<td>1 1 0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>1 1 1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>...</td>
</tr>
</tbody>
</table>
NFL Revisited (2/4)

- NFL shows that, averaged over all $f_1, f_2, \ldots, f_{256} \in \mathcal{F}$, the behavior on $Te$ of any learner trained on $Tr$ is that of a random guesser.
- This result is rather intuitive.
  - Consider functions $f_1$ through $f_4$.
  - For all 4 functions, $Tr$ is the same.
  - Given any deterministic learner $L$, the model induced by $L$ from $Tr$ is the same in all 4 cases.
  - Since the associated $Te$'s span all possible labelings of OTS, for any OTS instance any model will be correct for half the functions and incorrect for the other half.
  - Argument is easily repeated across all such subsets of 4 functions, giving the overall result.
NFL Revisited (3/4)

- NFL simply restates Hume’s famous conclusion about induction having no rational basis
  - There can be no demonstrative arguments to prove, that those instances, of which we have had no experience, resemble those, of which we have had experience. Thus not only our reason fails us in the discovery of the ultimate connexion of causes and effects, but even after experience has inform’d us of their constant conjunction, ’tis impossible for us to satisfy ourselves by our reason, why we shou’d extend that experience beyond those particular instances, which have fallen under our observation. We suppose, but are never able to prove, that there must be a resemblance betwixt those objects, of which we have had experience, and those which lie beyond the reach of our discovery.

- All other things being equal, given that all we see is Tr and its labeling, there is no rational reason to prefer one labeling of Te over another.
NFL Revisited (4/4)

- Crucial and most powerful contribution of NFL
  - Whenever a learning algorithm performs well on some function, as measured by OTS generalization, it must perform poorly on some other(s)
  - Hence, building decision support systems for what learning algorithm works well where becomes a valuable endeavor
Ultimate Learning Algorithm (1/8)

- Let $p^\Omega$ be the non-uniform probability distribution over the $f_i$’s induced by some process $\Omega$ that presents learning problems.
- Given a training set, a learning algorithm, $L$, induces a model, $M$, which defines a class probability distribution, $p$, over the instance space.
- An *Ultimate Learning Algorithm (ULA)* is a learning algorithm that induces a model $M^*$, such that:

$$\forall M' \neq M^* \ E(\delta(p^*, p^\Omega)) \leq E(\delta(p', p^\Omega))$$

where the expectation is computed for a given training/test set partition of the instance space, over the entire function space, and $\delta$ is some appropriate distance measure.
Finding a ULA consists of finding a learning algorithm whose induced models closely match our world’s underlying distribution of functions.
Cross-validation is regularly used as a mechanism to select among competing learning algorithms.
Cross-validation is also subject to the NFL theorem

- Easily seen from earlier illustration
  - $Tr$ does not change over $f_1$ through $f_4$, so cross-validation always selects the same best learner in each case
  - The original NFL theorem applies

- It follows that cross-validation cannot generalize and thus cannot be used as a viable way of building an ultimate learning algorithm
Ultimate Learning Algorithm (5/8)

- Other extreme: design one’s own algorithm
- Assumes that the universe is such that the tasks likely to occur are exactly those that I am interested in solving (don’t care about others)
- This is a possibility but:
  - Makes stronger assumptions than we might like
  - Is laborious, and
  - Is somewhat at odds with the philosophy of machine learning
- Metalearning as a viable alternative?
Ultimate Learning Algorithm (6/8)

- Metalearning = learning an estimate of $p^\Omega$
- Assumes that it is possible to gather training data at the metalevel to learn that estimate
Assumptions that must be made for metalearning are considerably more natural than those that must be made for manual algorithm design.
We all hold deep-rooted, intuitive notions of bizarre functions

- Harkening back to Hume, there is no rational reason for these beliefs, which of course is the “riddle” of induction
- However, implicit in Western thinking is that if we were to make only one assumption, it would have to be that induction is valid, i.e., that we can generalize from what we have seen to things we have yet to encounter.

This is the fundamental assumption of science as we practice it. It also is fundamental to our being able to live at ease in the world, not constantly worrying for example that the next time we step on a bridge it will not support our weight.
Practical Considerations

▶ When a designer introduces a novel classification algorithm, how does she position it in the existing algorithm landscape?
▶ When a practitioner is faced with a new task for which she seeks a high accuracy model, how does she know which algorithm to use?
Implications of NFL

- Two views of the world
  1. *Closed Classification World Assumption (CCWA)*
     - She assumes that all classification tasks likely to occur form some well-defined subset of the universe. As a designer, she shows that her novel algorithm performs better than others on that set. As a practitioner, she picks any of the algorithms that performs well on that set.
  2. *Open Classification World Assumption (OCWA)*
     - She assumes no structure on the set of classification tasks. As a designer, she characterizes as precisely as possible the class of tasks on which her novel algorithm outperforms others. As a practitioner, she has some way of determining which algorithm(s) will perform well on her specific task(s).
ML Community Subconscious

- Widely-used approach consisting in benchmarking algorithm against well-known repositories (e.g., UCI) tends to implicitly favor the CCWA

- Yet, there is no known characterization of “real-life” classification tasks
To compound the problem for practitioners, most efforts in algorithm design seem to share the same oblivious pattern:

1. They propose new algorithms that overcome known limitations. Yet, unless one accepts the CCWA, this simply shifts the original question of how to overcome the targeted limitations to the equally difficult question of determining what applications the proposed approach works well on.

2. They “promote” new algorithms on the basis of limited empirical results, leaving the burden of proof to the users. It is not trivial to know how well any new approach will generalize beyond the problems it has been tested against so far.
Impact on Users

- Large number of learning algorithms, with comparatively little insight gained in their individual applicability
- Users are faced with a plethora of algorithms, and without some kind of assistance, algorithm selection can turn into a serious road-block for those who wish to access the technology more directly and cost-effectively
- End-users often lack not only the expertise necessary to select a suitable algorithm, but also the availability of many algorithms to proceed on a trial-and-error basis
- And even then, trying all possible options is impractical, and choosing the option that “appears” most promising is likely to yield a sub-optimal solution
DM Packages

- Commercial DM packages consist of collections of algorithms wrapped in a user-friendly graphical interface
  - Facilitate access to algorithms, but generally offer no real decision support to non-expert end-users
- Need an informed search process to reduce the amount of experimentation while avoiding the pitfalls of local optima
- Informed search requires metaknowledge
- Metalearning offers a robust mechanism to build metaknowledge about algorithm selection in classification
- In a very practical way, metalearning contributes to the successful use of Data Mining tools outside the research arena, in industry, commerce, and government
Rice’s Framework

A problem $x$ in problem space $P$ is mapped via some feature extraction process to $f(x)$ in some feature space $F$, and the selection algorithm $S$ maps $f(x)$ to some algorithm $a$ in algorithm space $A$, so that some selected performance measure (e.g., accuracy), $p$, of $a$ on $x$ is optimal.
Framework Issues

The following issues have to be addressed:

1. The choice of $f$,
2. The choice of $S$, and
3. The choice of $p$.

$A$ is a set of base-level learning algorithms and $S$ is itself also a learning algorithm.

Making $S$ a learning algorithm, i.e., using metalearning, has further important practical implications about:

1. The construction of the training metadata set, i.e., problems in $P$ that feed into $F$ through the characterization function $f$,
2. The content of $A$,
3. The computational cost of $f$ and $S$, and
4. The form of the output of $S$.
Choosing Base-level Learners

► No learner is universal
► Each learner has its own area of expertise, i.e., the set of learning tasks on which it performs well
► Select base learners with complementary areas of expertise
► Two issues in this choice:
  ► Coverage
  ► Size
Good Coverage

- Seek the smallest set of learners that is most likely to ensure a reasonable coverage
- Not trivial
  - Experiments on the space of binary classification tasks of 3 Boolean variables
  - From 26 applicable algorithms, a subset of 7 is sufficient to obtain maximal coverage
  - But 9 tasks still remain uncovered (i.e., none of the learners is better than chance on these)
- Recommendation:
  - Choose base learners have different biases by choosing representatives from varied model classes
  - The more varied the biases, the greater the coverage
Nature of Training Metadata

- Challenge:
  - Training data at metalevel = data about base-level learning problems or tasks
  - Number of accessible, documented, real-world classification tasks is small
Building Training Metadata

- Two alternatives:
  - Augmenting training set through systematic generation of synthetic base-level tasks
  - View the algorithm selection task as inherently incremental and treat it as such

- Recommendation:
  - First approach is non-trivial and probably limited to well-defined data characteristics
  - Second approach naturally adapts to reality, extending to new areas of the base level learning space only when tasks from these areas actually arise
Meta-examples

- Meta-examples are of the form $< f(x), t(x) >$, where $t(x)$ represents some target value for $x$.
- By definition, $t(x)$ is predicated upon $p$, and the choice of the form of the output of $S$.
- Focusing on the case of selection of 1 of $n$: 
  $$t(x) = \arg\max_{a \in A} p(a, x)$$
- Metalearning takes $\{< f(x), t((x)) > : x \in P' \subseteq P\}$ as a training set and induces a metamodel that, for each new problem, predicts the algorithm from $A$ that will perform best.
- Constructing meta-examples is computationally intensive.
Choosing $f$

- As in any learning task, the characterization of the examples plays a crucial role in enabling learning
- Features must have some predictive power
- Four main classes of characterization:
  - Statistical and information-theoretic
  - Model-based
  - Landmarking
  - Learning Curves
Statistical and Information-theoretic Characterization

- Extract a number of statistical and information-theoretic measures from the labeled base-level training set
- Typical measures include number of features, number of classes, ratio of examples to features, degree of correlation between features and target, class-conditional entropy, skewness, kurtosis, and signal to noise ratio
- Assumption: learning algorithms are sensitive to the underlying structure of the data on which they operate, so that one may hope that it may be possible to map structures to algorithms
- Empirical results do seem to confirm this intuition
Model-based Characterization

- Exploit properties of a hypothesis induced on problem $x$ as an indirect form of characterization of $x$

- Advantages:
  1. Dataset is summarized into a data structure that can embed the complexity and performance of the induced hypothesis, and thus is not limited to the example distribution
  2. Resulting representation can serve as a basis to explain the reasons behind the performance of the learning algorithm

- To date, only decision trees have been considered, where $f(x)$ consists of either the tree itself, if the metalearning algorithm can manipulate it directly, or properties extracted from the tree, such as nodes per feature, maximum tree depth, shape, and tree imbalance
Landmarking (1/4)

- Each learner has an area of expertise, i.e., a class of tasks on which it performs particularly well, under a reasonable measure of performance.

- Basic idea of the landmarking approach:
  - Performance of a learner on a task uncovers information about the nature of the task.
  - A task can be described by the collection of areas of expertise to which it belongs.

- A *landmark learner*, or simply a *landmarker*, a learning mechanism whose performance is used to describe a task.

- Landmarking is the use of these learners to locate the task in the *expertise space*, the space of all areas of expertise.
Landmarking = finding locations of tasks in expertise space

- Assume that $i_1$, $i_2$, and $i_3$ are taken as landmarkers
- Problems on which both $i_1$ and $i_3$ perform well, but on which $i_2$ performs poorly, are likely to be in $i_4$'s area of expertise
Landmarking (3/4)

- Concentrate solely on cartographic considerations
- Exploring the metalearning potential of landmarking amounts to investigating how well a landmark learner’s performance hints at the location of the learning tasks in the expertise map
- In principle, every learner’s performance can signpost the location of a problem with respect to other learner’s expertise
- In practice, however, we want landmark learners to be efficient
Landmarking (4/4)

- The *prima facie* advantage of landmarking resides in its simplicity: learners are used to signpost learners.
- Need efficient landmarkers.
  - Use naive learning algorithms (e.g., OneR, Naive Bayes) or “scaled-down” versions of more complex algorithms (e.g., DecisionStump).
- Results with landmarking have been promising.
Partial Learning Curves (1/3)

- Learning curve = performance measure (e.g., accuracy) as a function of the size of the training set
- Partial learning curves may be used as an indirect kind of characterization to select between algorithms

Method:
- Training metadata consists of triplets $< D, l_{c_{A_1,D}}, l_{c_{A_2,D}} >$
  - $D$ is a (base-level) dataset
  - $l_{c_{A_1,D}}$ (resp., $l_{c_{A_2,D}}$) is the learning curve for $A_1$ (resp., $A_2$) on $D$, computed with progressive sampling
- Each learning curve is in turn represented as a vector $< a_{A_k,D,1}, a_{A_k,D,2}, \ldots, a_{A_k,D,#S} >$, where $a_{A_k,D,r}$ is the accuracy of algorithm $A_k$ on dataset $D$ for the $r$-th sample
The distance between two datasets $D_i$ and $D_j$, in the context of discriminating between the predictive performances of algorithms $A_1$ and $A_2$, is given by:

$$d_{A_1,A_2}(D_i, D_j) = \sum_{m=1}^{\#S} \left[ (a_{A_1,D_i,m} - a_{A_1,D_j,m})^2 + (a_{A_2,D_i,m} - a_{A_2,D_j,m})^2 \right]$$

$$= \sum_{k=1}^{2} \sum_{m=1}^{\#S} (a_{A_k,D_i,m} - a_{A_k,D_j,m})^2$$

Generalizing to $n$ learning algorithms and partial curves of varying lengths, the distance between $D_i$ and $D_j$ is:

$$d_{A_1,...,A_n}(D_i, D_j) = \sum_{k=1}^{n} \sum_{m=1}^{\#S_k} (a_{A_k,D_i,m} - a_{A_k,D_j,m})^2$$
Partial Learning Curves (3/3)

- When a new target dataset $T$ is presented
  - $A_1$ and $A_2$ are executed to compute their partial learning curves on it, up to some pre-defined sample size, $\#S$
  - The 3 nearest neighbors of $T$ are identified using the distance function $d$
  - The accuracies of $A_1$ and $A_2$ on these neighbors for sample size $|T|$ are retrieved (or computed), from the database
  - Each neighbor votes for either $A_1$, if $A_1$ has higher accuracy than $A_2$ at the target size, or $A_2$, if the opposite is true
  - The “best” algorithm predicted for $T$ corresponds to the majority vote

- Moderate success; more work needed here
Computational Cost

- Necessary price to pay to be able to perform algorithm selection learning at the metalevel
- To be justifiable, the cost of computing $f(x)$ should be significantly lower than the cost of computing $t(x)$
- The larger the set $A$ and the more computationally intensive the algorithms in $A$, the more likely it is that the above condition holds
- In all implementations of the aforementioned characterization approaches, that condition has been satisfied
- Cost of induction vs. cost of prediction (batch vs. incremental)
Selecting on Accuracy

- Predictive accuracy has become the *de facto* criterion, or performance measure
- Bias largely justified by:
  - NFL theorem: good performance on a given set of problems cannot be taken as guarantee of good performance on applications outside of that set
  - Impossibility of forecasting: cannot know how accurate a hypothesis will be until that hypothesis has been induced by the selected learning model and tested on unseen data
  - Quantifiability: not subjective, induces a total order on the set of all hypotheses, and straightforward, through experimentation, to find which of a number of available models produces the most accurate hypothesis
Selecting on Other Criteria (1/2)

- There are a number of other aspects which have an impact on which model to select
- In fact:
  - Empirical evidence suggests that for large classes of applications, most learners perform well in terms of accuracy
  - Yet they often exhibit extreme variance along other dimensions
- This (ab)use of accuracy is also a result of the assumption that predicting data values equates to predicting useful business outcomes
  - Again, evidence suggests that this assumption may not always hold and other factors should be considered
Selecting on Other Criteria (2/2)

- Other performance measures:
  - Expressiveness
  - Compactness
  - Computational complexity
  - Comprehensibility
  - Etc.

- These could be handled in isolation or in combination to build multi-criteria performance measures

- To the best of our knowledge, only computational complexity, as measured by training time, has been considered in tandem with predictive accuracy
Selection vs. Ranking

- **Standard**: single algorithm selected among \( n \) algorithms
  - For every new problem, metamodel returns one learning algorithm that it predicts will perform best on that problem

- **Alternative**: ranking of \( n \) algorithm
  - For every new problem, metamodel returns set \( A_r \subseteq A \) of algorithms ranked by decreasing performance
Advantages of Ranking

- Ranking reduces brittleness
- Assume that the algorithm predicted best for some new classification problem results in what appears to be a poor performance
  - In the single-model prediction approach, the user has no further information as to what other model to try
  - In the ranking approach, the user may try the second best, third best, and so on, in an attempt to improve performance
- Empirical evidence suggests that the best algorithm is generally within the top three in the rankings
Metalearning-inspired Systems

- Although a valid intellectual challenge in its own right, metalearning finds its real raison d’être in the practical support it offers Data Mining practitioners.
- Some promising implementations:
  - MininMart
  - Data Mining Advisor
  - METALA
  - Intelligent Discovery Assistant
- Mostly prototypes, work in progress
MiningMart

- Algorithm selection for preprocessing
- Goal is to enable reuse of successful preprocessing phases across applications through CBR
  - Capture information about both data and operator chains through metamodel (M4) and computer interface
  - Case: complete description of a preprocessing phase in M4
  - New mining task: user searches through MiningMart’s case base for the case that seems most appropriate
  - Once a useful case has been located, it can be downloaded
  - The local version of the system then generates preprocessing steps that can be executed automatically for the current task
- http://mmart.cs.uni-dortmund.de/end-user/caseBase.html
Data Mining Advisor (1/2)

- Algorithm ranking for model building (classification)
- Given a dataset and user-defined goals for accuracy and training time, returns a list of algorithms ranked according to how well they meet the stated goals
- Base-level algorithms:
  - Decision trees: C5.0rules, C5.0tree and C5.0boost
  - Linear models: linear tree (ltree), linear discriminant (lindiscr)
  - Instance-based: MLC++ IB1 (mlcib1)
  - Probability-based: Naïve Bayes (mlcnn)
  - Neural networks: SPSS Clementine’s Multilayer Perceptron (clemMLP), RBF Networks (clemRBFN)
  - Rule-based: Ripper
Wizard-like step-by-step process:

1. Upload dataset
2. Characterize dataset (statistical and information-theoretic measures)
3. Parameter setting and ranking
   - Selection criteria: 3 predefined trade-off levels between accuracy and training time
   - Ranking method: 2 ranking mechanisms
4. Execute (currently disabled)
   - Select any number of algorithms
   - Return 10-fold CV accuracy, true rank and score, and, when relevant, training time

http://www.metal-kdd.org
Agent-based architecture for distributed Data Mining, supported by metalearning

Aim is to provide a system that:

1. Supports an arbitrary number of algorithms and tasks (i.e., $P$, and more importantly $A$ may grow over time)
2. Automatically selects an algorithm that appears best from the pool of available algorithms, using metalearning

Algorithm characterized by features relevant to its usage

- Type of input data
- Type of induced model
- How well noise is handled
- Etc.
Each learning algorithm is embedded in an agent that provides clients with a uniform interface to three basic services: configuration, model building and model application.

Each task is characterized by statistical and information-theoretic features, as in the DMA.

Designed to autonomously and systematically carry out experiments with each task and each learner and, using task features as meta-attributes, induce a metamodel for algorithm selection.

As new tasks and algorithms are added, corresponding experiments are performed and the metamodel is updated.
No metalearning yet, but...

Unique in that, unlike the previous systems, it encompass the three main algorithmic steps of the KDD process (preprocessing, model building and post-processing)

Any chain of operations consisting of one or more operations from each of these steps is called a Data Mining (DM) process

Goal of IDA: propose list of ranked DM processes that are both valid and congruent with user-defined preferences
Intelligent Discovery Assistant (2/5)

- Underlying ontology/taxonomy of DM operations or algorithms, where leaves represent implementations available in the corresponding IDA

\[ \text{rs} = \text{random sampling (10%)}, \quad \text{fbd} = \text{fixed-bin discretization (10 bins)}, \quad \text{cbd} = \text{class-based discretization}, \quad \text{cpe} = \text{CPE-thresholding post-processor} \]
Intelligent Discovery Assistant (3/5)

- Operations characterized by pre-conditions, post-conditions and heuristic indicators
- Plan generator
  - Input: dataset, user-defined objective (e.g., build a fast, comprehensible classifier) and user-supplied information about the data (that may not be obtained automatically)
  - Start with an empty process
  - Search for operation whose pre-conditions are met and whose indicators are congruent with user-defined preferences
  - Once an operation has been found, it is added to the current process, and its post-conditions become the system’s new conditions from which the search resumes
  - Search ends once a goal state has been reached or when it is clear that no satisfactory goal state may be reached
Exhaustive search: all valid DM processes are computed

E.g.: continuous-valued data, preference on comprehensibility

<table>
<thead>
<tr>
<th>Plan #</th>
<th>Steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plan #1</td>
<td>C4.5</td>
</tr>
<tr>
<td>Plan #2</td>
<td>PART</td>
</tr>
<tr>
<td>Plan #3</td>
<td>rs, C4.5</td>
</tr>
<tr>
<td>Plan #4</td>
<td>rs, PART</td>
</tr>
<tr>
<td>Plan #5</td>
<td>fbd, C4.5</td>
</tr>
<tr>
<td>Plan #6</td>
<td>fbd, PART</td>
</tr>
<tr>
<td>Plan #7</td>
<td>cbd, C4.5</td>
</tr>
<tr>
<td>Plan #8</td>
<td>cbd, PART</td>
</tr>
<tr>
<td>Plan #9</td>
<td>rs, fbd, C4.5</td>
</tr>
<tr>
<td>Plan #10</td>
<td>rs, fbd, PART</td>
</tr>
<tr>
<td>Plan #11</td>
<td>rs, cbd, C4.5</td>
</tr>
<tr>
<td>Plan #12</td>
<td>rs, cbd, PART</td>
</tr>
<tr>
<td>Plan #13</td>
<td>fbd, NB, cpe</td>
</tr>
<tr>
<td>Plan #14</td>
<td>cbd, NB, cpe</td>
</tr>
<tr>
<td>Plan #15</td>
<td>rs, fbd, NB, cpe</td>
</tr>
<tr>
<td>Plan #16</td>
<td>rs, cbd, NB, cpe</td>
</tr>
</tbody>
</table>
Once all valid DM processes have been generated, heuristic ranker is applied to organize processes in descending order of “return” on user-specified goals.

- Ranking relies on knowledge-based heuristic indicators.
- For example, the processes above are ordered from simplest (i.e., least number of steps) to most elaborate.
- If speed rather than simplicity were the objective then Plan #3 would be bumped to the top of the list, and all plans involving random sampling (rs operation) would also move up.
- Currently rankings rely on fixed heuristic mechanisms.
  - However, IDAs are independent of ranking method and, so, could be improved by incorporating metalearning to generate rankings based on past performance.
Not metalearning, but...

Addresses one of the main problems: training metadata

Build a database to collect and organize all data relevant to machine learning experiments

- Data characteristics
- Algorithm parameter settings
- Algorithm properties
- Performance measures
- Etc.
 Experiment Database (2/2)

- The database:
  - Is extendible
  - Is public
  - Contains over 650,000 experiments

- Tremendous contribution to the community and a serious boost to metalearning research
  - Can be used to produce new information, to test hypotheses, to verify existing hypotheses or results, and to perform experiment mining

Work in Progress

- Metalearning is still a relatively young area of research
- Efforts so far have demonstrated promise
- Much room for improvement as well as the development of new ideas and systems
  - Characterizing learning algorithms and gaining a better understanding of their behavior
  - Defining and effectively operationalizing multi-criteria performance measures
  - Designing of truly incremental systems, where new problems and new (base-level) algorithms may be continually added without retraining the system
Getting Involved

- **New book**

- **Survey**

- **Google Group**
  - [http://groups.google.com/group/meta-learning](http://groups.google.com/group/meta-learning)