Meta-Learning

Pavel Brazdil

LIAAD - INESC TEC / FEP, Univ. of Porto

April 2013

Overview

1 The Algorithm Selection Problem

2 Employing Meta-Learning for the Selection of Classification Algorithms

- 2.1 Identifying Suitable Algorithms with Meta-Level Models
- 2.2 Acquisition of Meta-Knowledge
- 2.3 Dynamic / Iterative Approaches
  - Combining algorithm selection and characterization via testing

3 Employing Meta-Learning in KDD Workflow Design

4 Meta-learning for model selection in other domains

1. Algorithm / Model Selection Problem

Typically many different algorithms exist in a particular domain (classification, regression, optimization etc.).

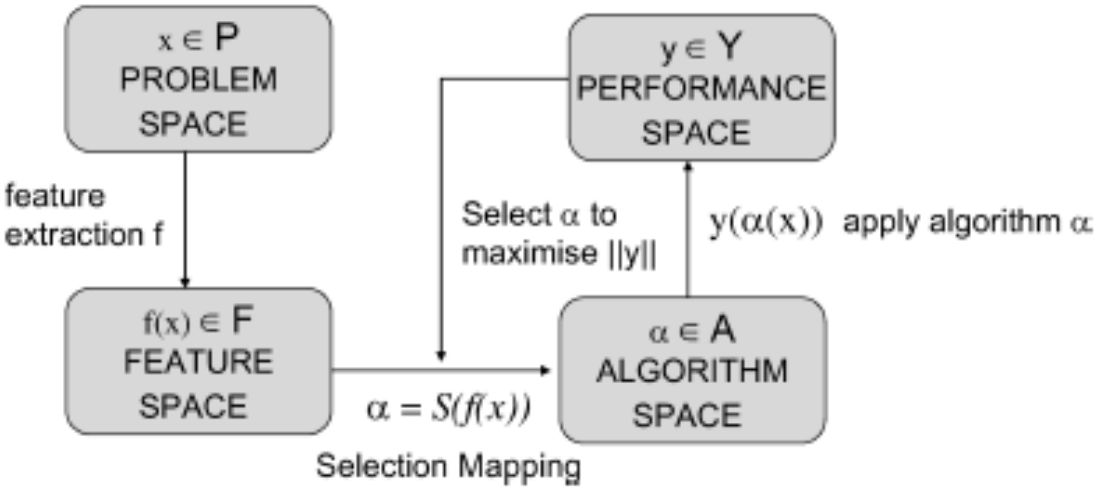
We want methods that can help us to select the one with the best performance.

This problem was first formulated by Rice [1976]:

For a given problem instance $x \in P$, with features $f(x) \in F$, find the selection mapping $S(f(x))$ into algorithm space A, such that the selected algorithm $\alpha \in A$ maximizes the performance mapping $y(\alpha(x)) \in Y$.

Algorithm / Model Selection Problem

Schema suggested by Rice [1976]:



In subsequent work the selection mapping S(f(x)) was generated using ML methods.
The process is often referred to as meta-learning.

The process can be applied to the problem of selecting classification algorithms (see next slides).

2. Employing Meta-Learning for the Selection of Classification Algorithms

A large set of techniques is available in Machine Learning (ML).
+ It increases a possibility that a good solution can be found.
- It is much harder to find the right ML algorithm,
as many alternatives exist.

The problem of selecting a suitable (the best) algorithm can be seen as a problem of search.

We cannot test all ML algorithms for computational reasons (there are thousands of variants of ML algorithm + parameter settings)

Why meta-learning?
It helps to build on previous experience and identify the right algorithm more effectively.

In this part we focus on classification algorithms.

Different approaches

2.1 Identifying Suitable Algorithms with Meta-Level Models

2.2 Acquisition of Meta-Knowledge

2.3 Dynamic / Iterative Approaches

Combine algorithm selection and characterization via testing

References:

P.Brazdil, C.Giraud-Carrier, C.Soares, R.Vilalta: Metalearning: Applications to Data Mining, Springer, 2009

K.Smith-Miles: Cross-Disciplinary Perspectives on Meta-Learning for Algorithm Selection, ACM Computing Surveys, 2008

2.1 Identifying Suitable Algorithms with Meta-Level models

Phase 1:

Consider the given new dataset,
construct characteristics / meta-features.

Exploit meta-level model to
identify a suitable subset of algorithms.

In some work the result is a ranked subset of classification algorithms permitting reduced search.

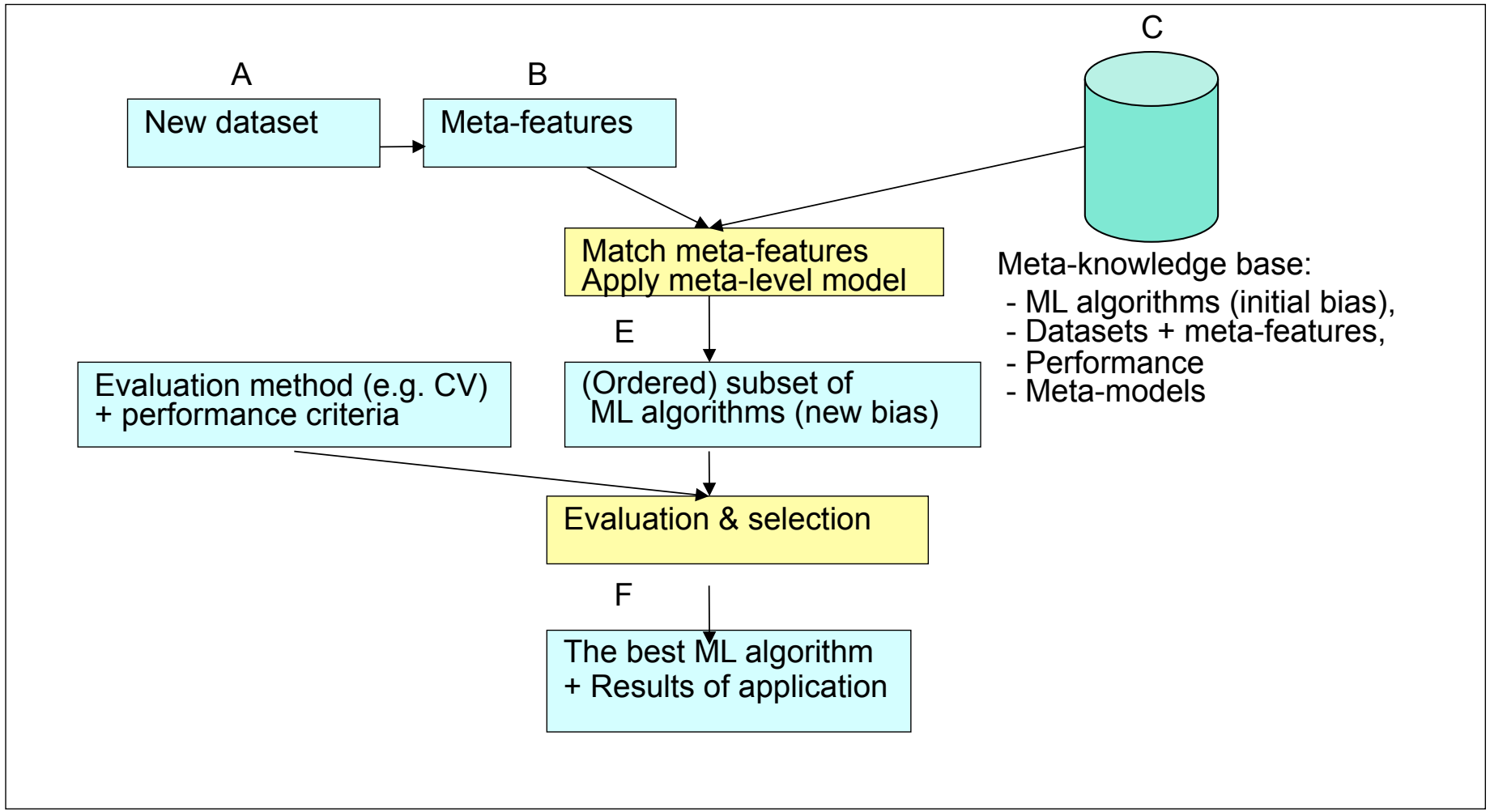
Phase 2: Search through the reduced space of algorithms.

Evaluate each option using
a chosen evaluation method (typically a cross-validation) and
a given performance criteria (e.g. accuracy).

Identify the best alternative (or algorithms that are comparable).

Selecting ML Algorithms on Meta-features

Fig. 1.1



Data Characteristics / Meta-Features

The selection method relies on dataset characteristics or meta-features to provide some information that can differentiate performance of a set of given learning algorithms.

These typically include :

- **statistical and information-theoretic measures,**
- **model based characterization,**
- **landmarking,**
- **sub-sampling landmarks.**

Statistical and Information-Theoretic Measures

These measures typically include :

- number of classes,
- number of features,
- ratio of examples to features,
- degree of correlation between features and target concept,
- average class entropy
etc.

+ Positive and tangible results (e.g., ESPRIT Statlog and METAL).

- There is a limit on how much information these measures can capture, as these measures are uni- or bi-lateral measures only (capture relationships between two attributes only or one attribute and the class).

Landmarks

Examine performance of a set of simple and fast learning algorithms (landmarkers).

The accuracy of these landmarkers is used to characterize the dataset.

Sub-Sampling Landmarks

Exploit information obtained
on simplified versions of the data (samples).

Accuracy results on these samples serve
to characterise individual datasets and
are referred to as *sub-sampling landmarks*.

This information can be used
to select an appropriate learning algorithm.

One variant of the method uses
performance results on small samples s_1 .. s_k
to identify a (similar) learning curve and
estimate performance for sample s_N (whole dataset).
The algorithm with the highest performance at s_N is chosen.

Using a Meta-level Model to Select a Classification Algorithm

A meta-level system / model helps to
map characteristics into classification algorithms.

The meta-level system / model can be in the form of:
- **meta-level rules,**
- **k-NN (on the meta-level),**
- **neural network,**
- **other type of classification model on the meta-level.**

Employing a Rule Model on the Meta-Level

Early approaches (Rendell & Cho, 1990) used rules, such as:

- If the given dataset characteristics are C1, C2 .., Cn
then use algorithm A1 in preference to algorithm A2.

Ex.

IF (# training instance < 737) AND
(# prototype per class > 5.5) AND
(# relevants > 8.5) AND ***(relevant features)***
(# irrelevants < 5.5)
THEN IB1 will be better than CN2

Rules of this type can be used to identify a subset of algorithms to be evaluated.

Employing 1-NN on the Meta-Level

One simple approach uses 1-NN :

- **Compare meta-level characteristics of the new problem with meta-level characteristics of past problems,**
- **Identify the most similar dataset**
- **Retrieve either :**
  - *** The classification algorithm that performed best on that dataset,**
  - *** Ranking of classification algorithms, ordered by performance.**

Employing k-NN on the Meta-Level

A more complex approach employs k-NN:

- **Uses k-NN method to identify the most similar datasets.**
- **For each of these datasets,**
  retrieves the ranking of the candidate classification algorithms,
  based on past performance criteria (accuracy, learning time).
- **Aggregate the rankings obtained**
  to generate the final recommended ranking of algorithms.

Different Types of Output of Meta-Models

The meta-models can output:

- Just one algorithm considered applicable
Disadvantage: there may be other applicable algorithms

- A subset of algorithms considered applicable
(a subset of the given set),
Disadvantage: no information as to which one we should use first

- Ordered (sub)set of algorithms (ranking),

- Ordered (sub)set of algorithms (ranking) accompanied by the predictions of performance
Disadvantage: Predicting the values of performance may be difficult.

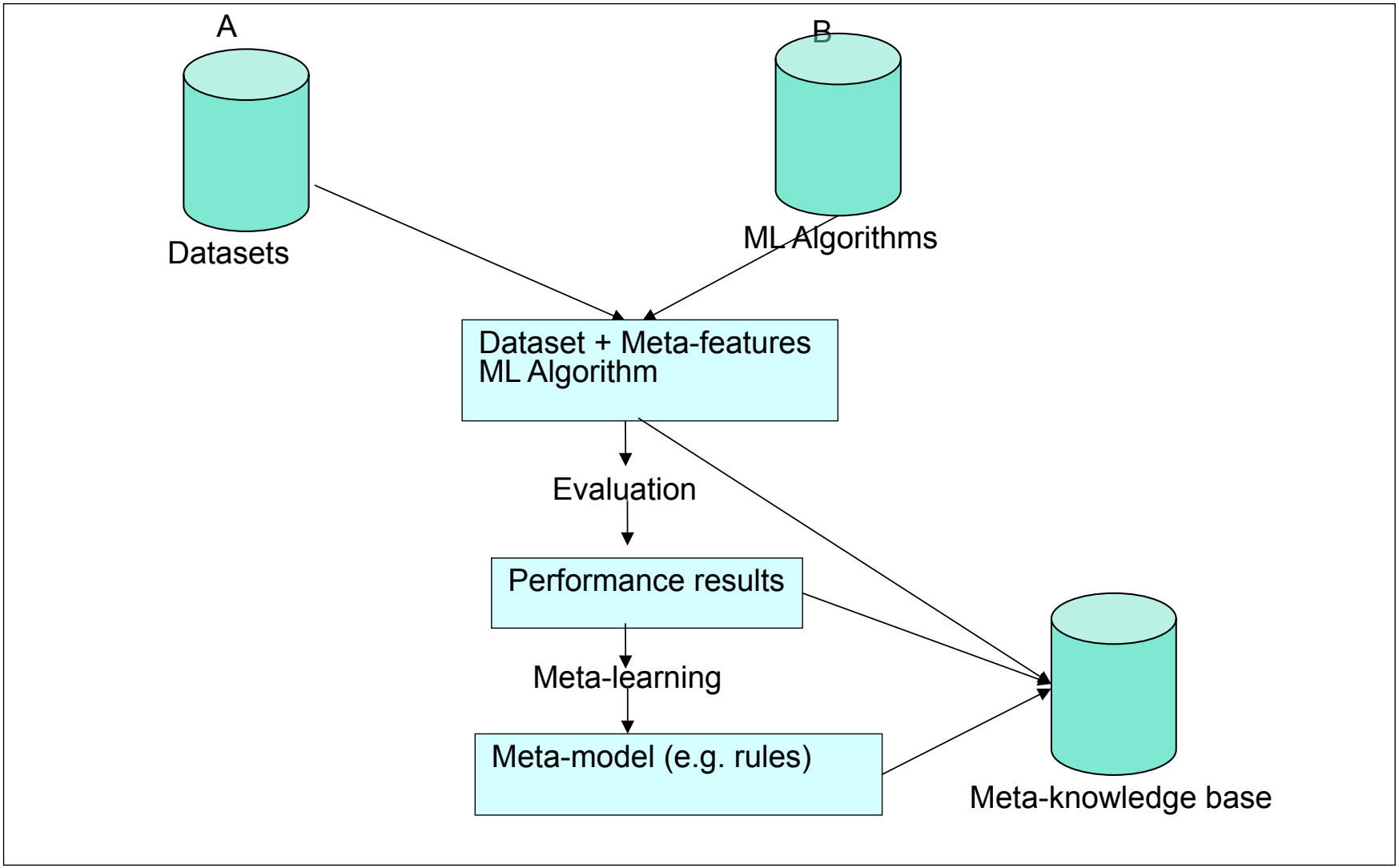
Phase II:
Evaluate the top N elements in the ordered (sub)set of algorithms (ranking) and select the one with the best performance.

2.2 Acquisition of Meta-Knowledge

- **Hand-code the rules, on the basis of on expert's knowledge:**
  - **rules are likely to be incomplete,**
  - **problem of maintenance;**
- **Automatic way of generating rules or models (see next figure):**
  - **Identify a set of problems (datasets),**
    set of classification algorithms (operations),
    define the evaluation method (D),
  - **Carry out experiments and store performance results (Fig. F),**
  - **Extend / reorganize the meta-knowledge:**
    - Either just store the meta-data for a lazy learning method (e.g. k-NN) or
    - Generate meta-level model (e.g. meta-rules)
    on the bases of meta-data (the model generalizes meta-data)

Acquisition of Meta-Knowledge

.



2.3 Dynamic and Iterative Approaches

Dynamic / iterative approaches combine:

- **Search for some initial set of suitable algorithms,**
- **Identifying alternative algorithms,**
- **Testing of the alternatives identified,**
- **Taking the test results into account in further search,**
- **Repeating this process**

while exploiting meta-level information.

The aim is to identify the best algorithm for some dataset d_{new} using a small number of tests.

Iterative Approach of Identifying the Best ML Algorithm

The search for the initial set of suitable algorithms

can identify just one algorithm,

the current best candidate algorithm a_{best}.

Identification of the alternatives can lead to just one algorithm at a time,

the best competitor a_k of a_{best}.

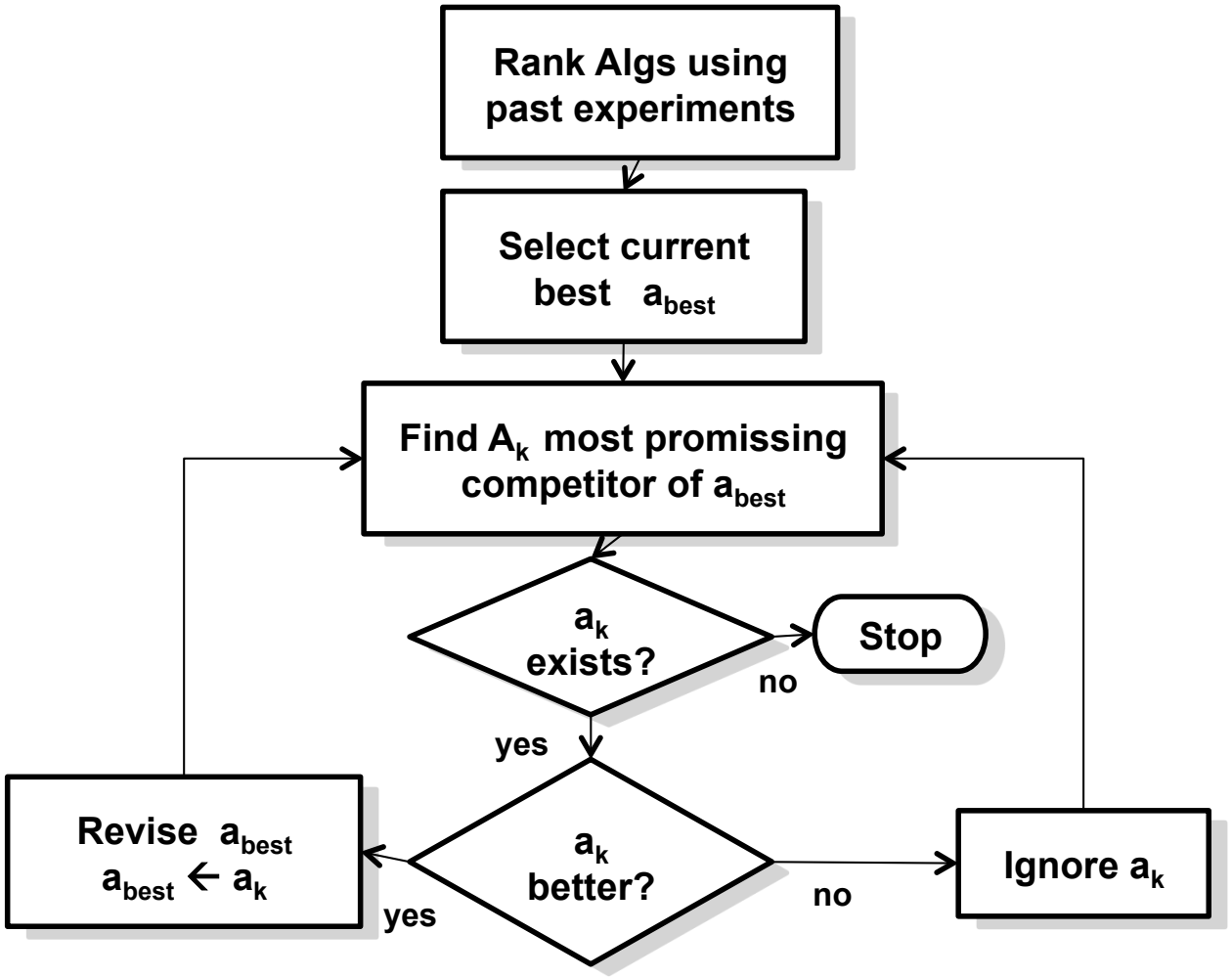
The details of a method based on these concepts are given in the following.

References:

R.Leite, P.Brazdil, J. Vanschoren: Selecting Classification Algorithms with Active Testing, Proc. of MLDM, Springer, 2012;

R.Leite, P.Brazdil, J. Vanschoren: Selecting Classification Algorithms with Active Testing on Similar Datasets, Proc. of PlanLearn-2012

Iterative Approach



Identifying the Best Candidate Algorithm

How to identify the initial best candidate algorithm a_{best}?

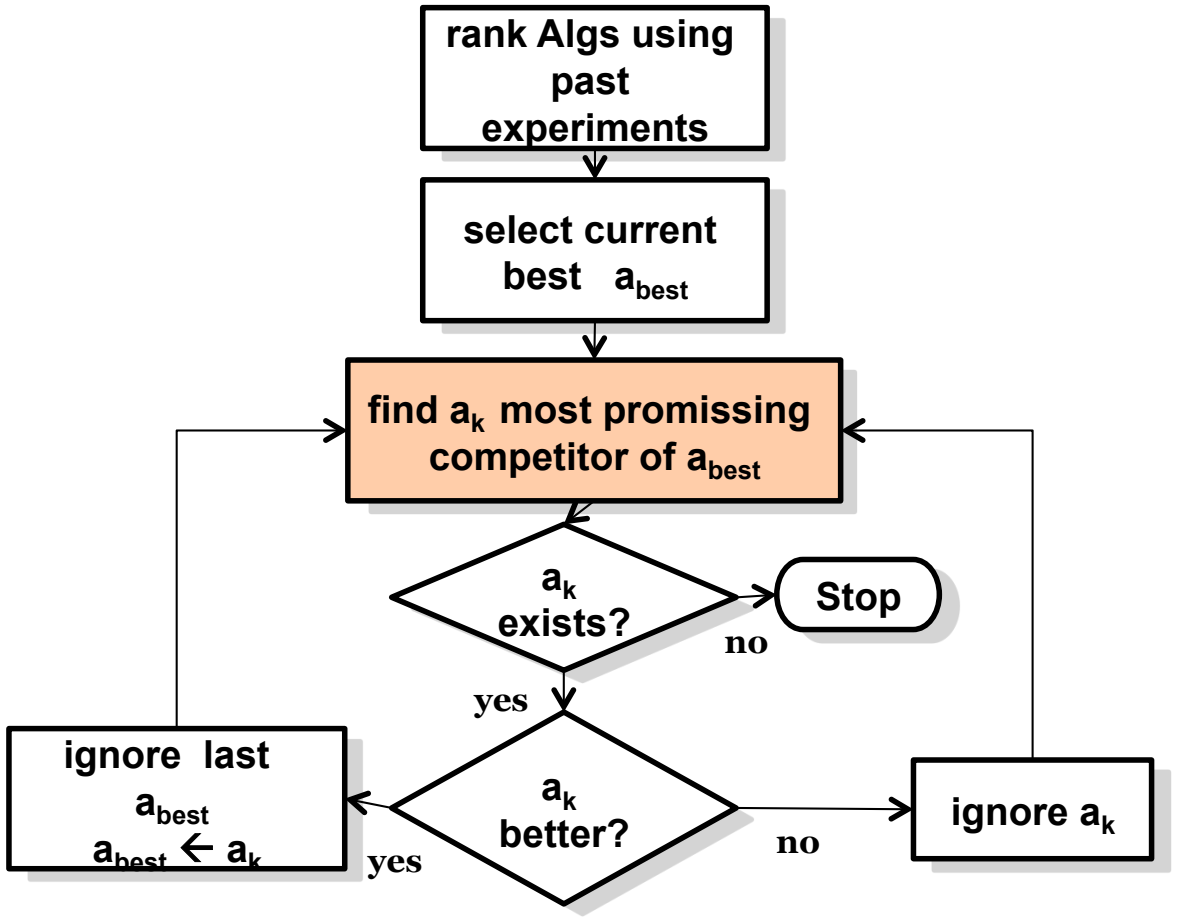
Use the algorithm that had the best performance overall in the past.

One method:

Retrieve performance results on datasets used in the past;
Calculate ranks of the algorithms for each dataset;
Calculate mean rank of each algorithm;
a_{best} is the algorithm with the lowest mean rank.

Datasets	IB1	J48	JRip	···
abalone	.197 (5)	.218 (4)	.185 (6)	···
acetylation	.844 (1)	.831 (2)	.829 (3)	···
adult	.794 (6)	.861 (1)	.843 (3)	···
...	...	...	...	...
Mean rank	4.05	2.73	3.17	···

Identify the most promising competitor



Identify the most promising competitor

The aim is to identify the most promising competitor for the new dataset.

Method:

- Calculate the performance gain of each a_i with respect to a_{best} for each dataset used in the past,

- Take dataset similarity into account,

- Provide an aggregated measure of the possible performance gains for each a_i,

- Select a_i with highest aggregated measure of performance gain and use it as the best competitor.

Identify the algorithm with highest performance gain

- Calculate the performance gain, RL (relative landmark), for each a_i with respect to a_{best} for each dataset d_j used in the past:

$$RL(a_i, a_{best}, d_j) = i(M(a_i,d_j) > M(a_{best},d_j)) * M(a_i,d_j) - M(a_{best},d_j)$$

(only positive gains are considered)

$M(a_i,d_j)$ – measure of performance of algorithm a_i on dataset d_j

- Provide an aggregated measure of the performance gains for each a_i, Identify algorithm a_k with the highest aggregated performance gain

$$\arg\max_{a_i} \sum_{dj \in D} RL(a_i, a_{best}, d_j) * Sim(d_{new}, d_j))$$

where $Sim(d_{new},d_j)$ represents the dataset similarity.

Calculating dataset similarity

Calculating dataset similarity $Sim(d_{new},d_j)$:

*** Method based on meta-data characteristics**

Dataset d_j is similar to d_{new} if both have similar meta-data characteristics

*** Method based on performance meta-data**

Dataset d_j is similar to d_{new} if the relative performance on both has a similar pattern.

A part of the pattern are cases like these:

$M(a_i,d_{new}) > M(a_{best},d_{new})$ and $M(a_i,d_j) > M(a_{best},d_j)$

recent tests on d_{new} ***previous tests on d_j* used in the past**

M(..) = performance measure.

Performance-based similarity lead to better results than the similarity based on data characteristics.

Final steps of the iterative process

Final steps of the iterative process:

- **Conduct evaluation of a_k (CV),**
- **Determine whether a_k has a better performance than a_{best} ,**
  If it has, $a_{best} \leftarrow a_k$
- **Repeat iterating until the stopping criterion is established.**

Evaluation and Results

The number of possible classification algorithms, parameterized variants and ensembles was 292.

The aim was to identify the best one using a small number of tests.

Various variants of the method were studied (ATWs_k, etc.).

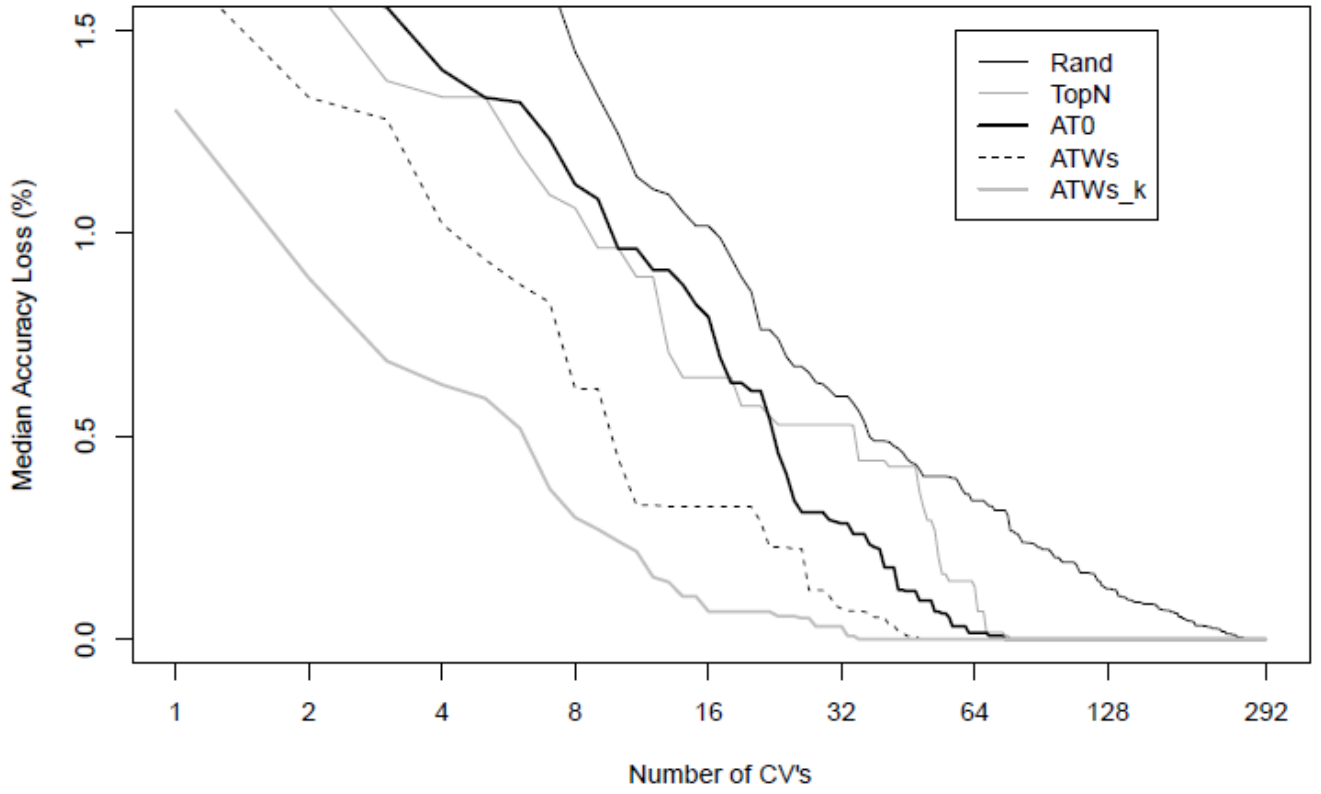
They achieve better performance than the baseline variants.

Baseline variants used for comparison:

- **Rand**
  Selects N algorithms at random.
  Uses CV to evaluate and select the best one.
- **TopN**
  Uses the mean rank on all datasets to select the topmost N algorithms
  Uses CV to evaluate and select the best one.

Evaluation and Results

Mean accuracy loss on the y axis (difference from the best accuracy) as a function of **number of CV tests** carried out (x axis).



3. Employing Meta-Knowledge in KDD Workflow Design

Overview.

3.1 What is a Workflow

3.2 Retrieving / Constructing Workflows

3.1 What is a Workflow / Plan?

The KDD process is typically represented in the form of a sequence of operations, such as :

- data selection,
- pre-processing,
- data mining / model generation,
- post-processing etc.

These can be represented either as simple sequences or as partially ordered a cyclical graphs.

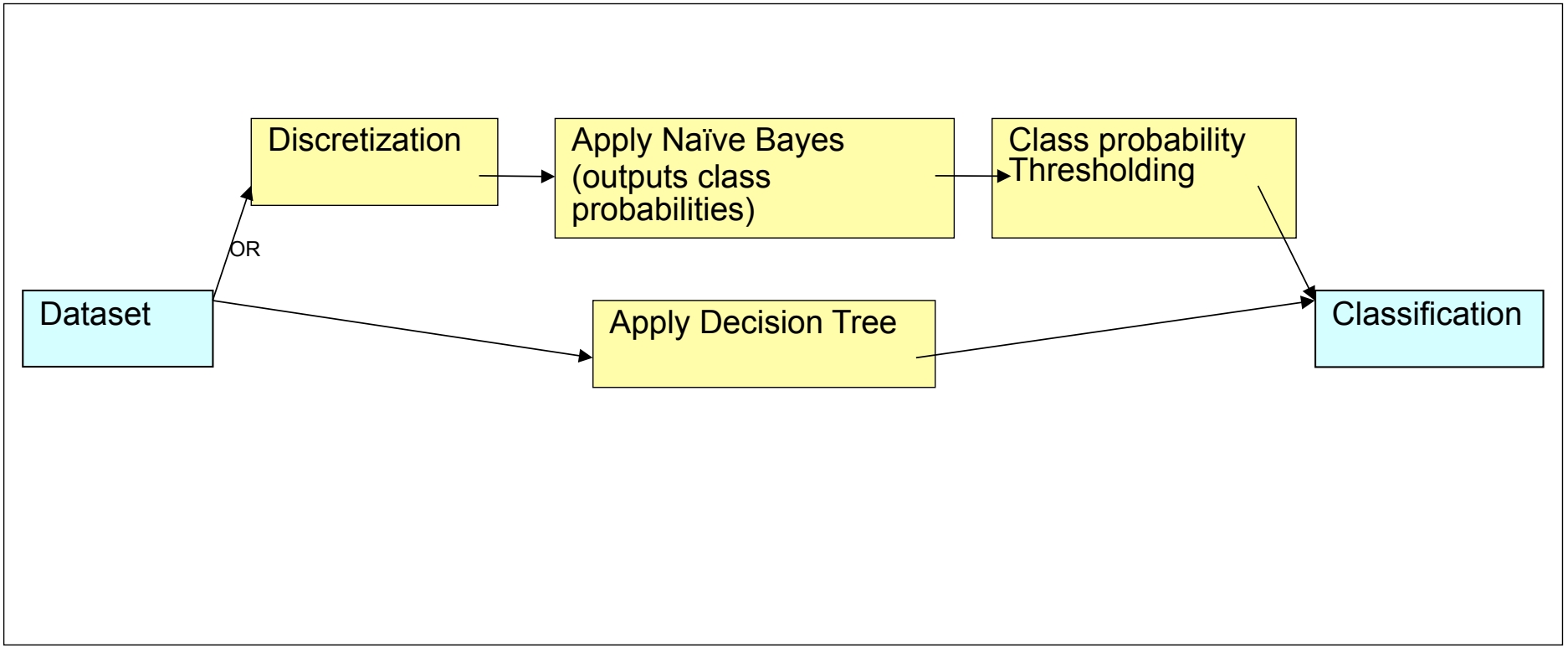
Bernstein and Provost call them *DM processes*.

An example of a simple partial order of operations is shown in the next figure.

Each partial order can be regarded as a workflow / plan to be executed. It will produce certain effects (e.g. classification of given instances).

Example of a Workflow / Plan

Example of a partial order of operations (workflow / plan)

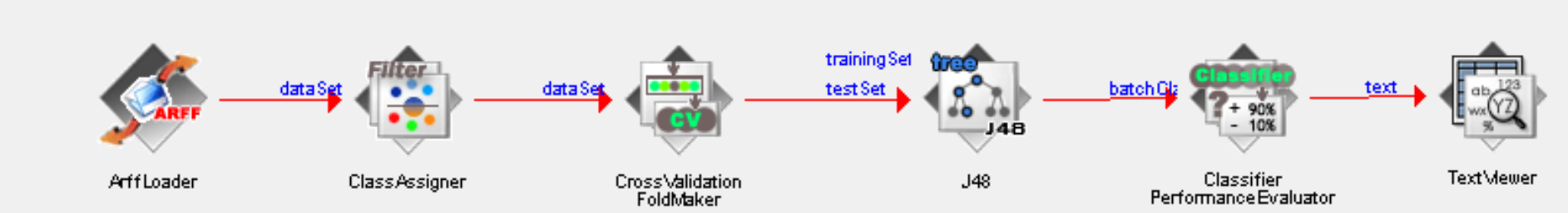


Workflows in DM Systems

Many DM Systems support workflows:

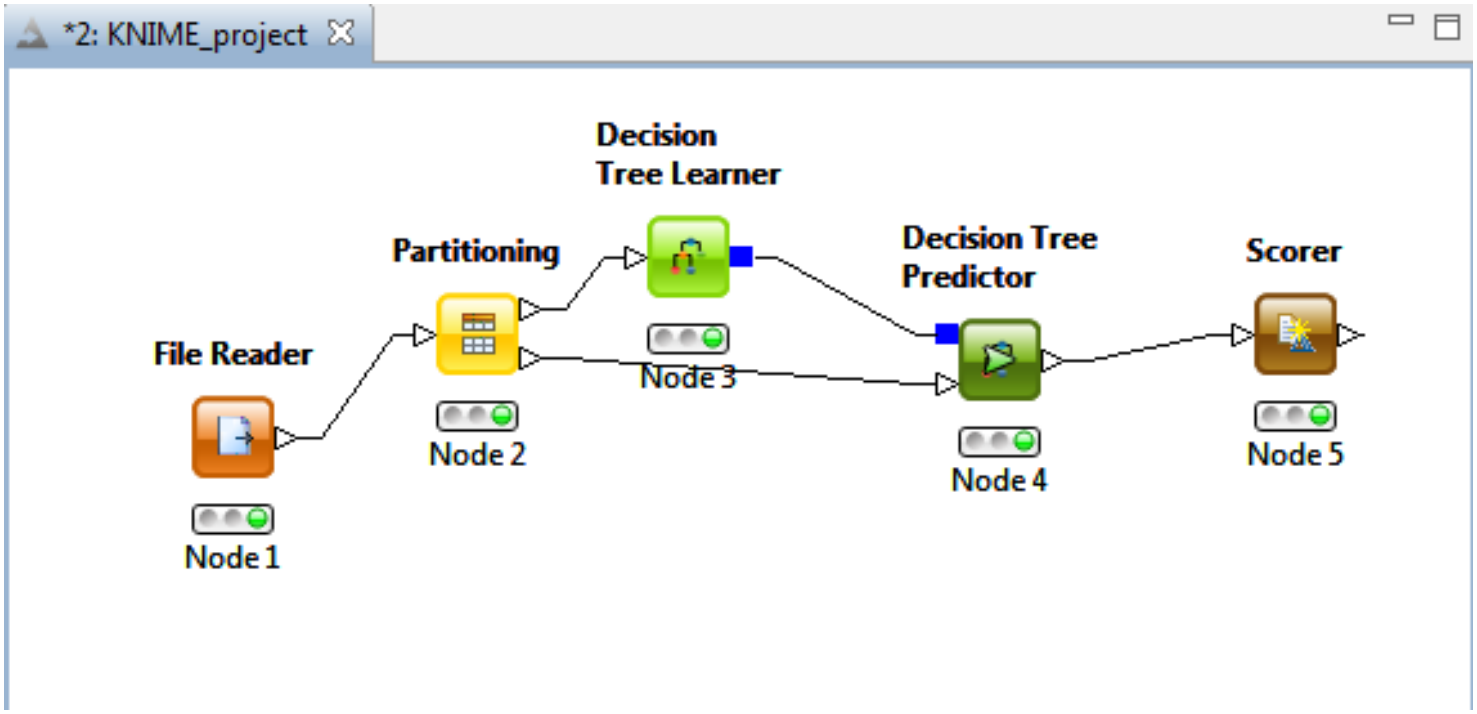
- SAS
- Weka
- Knime
- RapidMiner

Example of a Workflow in Weka:



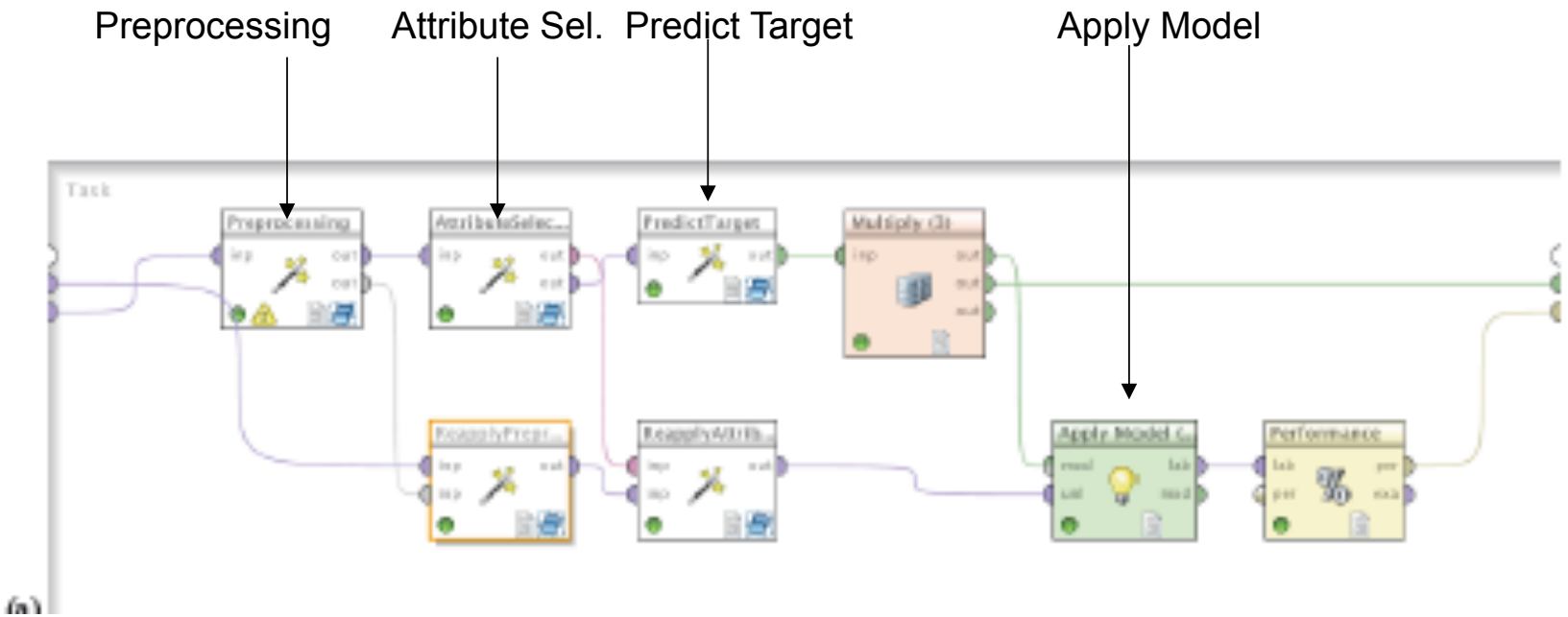
Workflows in DM Systems

Example of a Workflow in Knime:



Workflows in DM Systems

Example of a Workflow in RapidMiner:



3.2 Retrieving / Constructing Workflows

Workflows can be obtained in various ways by:

- Retrieving a suitable existing workflow and applying it,

- Similar to above, but permits manual or automatic modification of existing workflows (re-planning),

- Constructing a workflow by planning,

Retrieving a Suitable Workflow / Plan

Retrieving a suitable existing workflow / plan and applying it

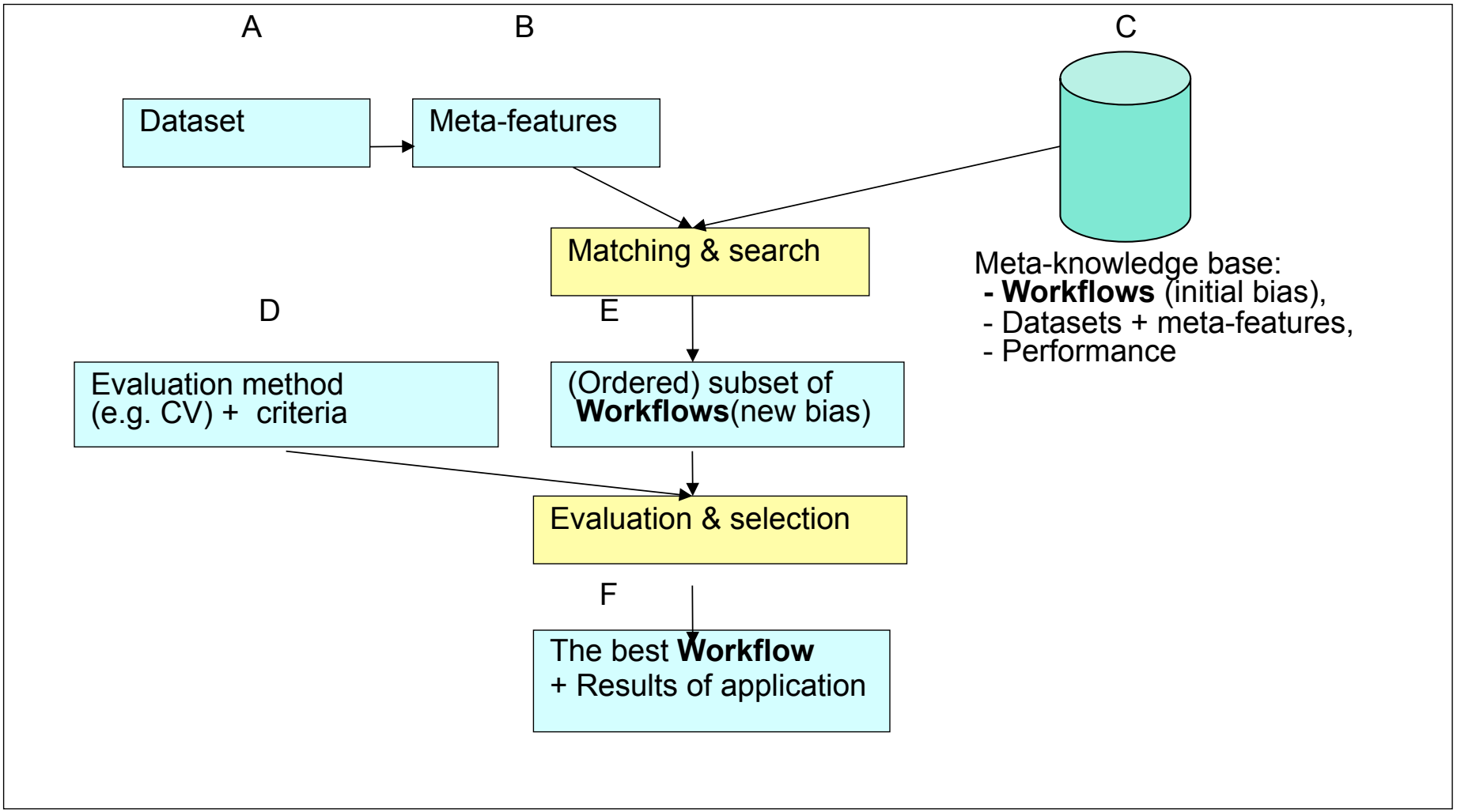
The method involves:

- **Considering previous workflows stored in meta-knowledge base,**
- **Matching the problem characteristics of the current problem with the characteristics of the past problems / workflows,**
- **Identifying a subset of potentially suitable workflows for the current problem,**
- **Evaluating the elements of the subset to identify the suitable one,**
- **Applying the workflow to the current problem.**

See the next figure

The existing workflow can be seen as embodying procedural meta-knowledge about the compositions of operations proved to be useful in the past.

Retrieving a suitable workflow from meta-database



Role of Meta-knowledge

Meta-knowledge can also capture the knowledge regards the applicability of existing workflows.

The existing workflow can be seen as embodying procedural meta-knowledge about the compositions of operations proved to be useful in the past.

Extending a Workflow / Plan

Problem of simple retrieval:
The meta-knowledge base may not contain any workflow that is suitable for the current problem.

So one possibility is to try to modify / adapt the retrieved workflow to the current situation.
This can be done manually or with the aid of planners.
The operation amounts to re-planning.

This method was used in MiningMart but the adaptation step was not automated.

Here we not give more details on this.

Constructing a Workflow / Plan

Constructing workflow / plan by planning:

The workflow / plan is built up from individual constituents.

The method start from an empty workflow / plan and gradually extends it, by adding operations to it.

Producing good workflows / plans is a non-trivial task.

It involves identifying a suitable set of operations, partial order among them so as to satisfy certain constraints and/or maximize certain evaluation measures.

The more operations we have, the more difficult it is.

For this reason, recent proposals resort to hierarchical planning.

Constructing a Workflow / Plan

This part is based on:

Kietz J-U, F.Serban, A.Bernstein, S.Fischer:
Designing KDD-Workflows via HTN-Planning, PlanLearn-2012.

In the following we review the method that exploits:

- **Planning knowledge**
- **Meta-data do describe I/O objects**
- **The task / method decomposition,**
- **operator models,**
- **probabilistic ranking of workflows.**

Planning Knowledge

The DM planning problem is modeled as DM Workflow ontology (DMWF).
It contains:
- **description of I/O objects and their meta data.**
- **tasks/operators from RapidMiner,**
- **methods (sequences of steps),**

The operators are structured in an inheritance hierarchy.
The operators include conditions and effects.
The inheritance hierarchy is supported by an ontology editor (eProPlan plugin to ontology editor Protégé).

Planning Knowledge

Many plans achieve the given goal (e.g. perform regression).

Plans / workflows are characterized by costs (function of associated error).

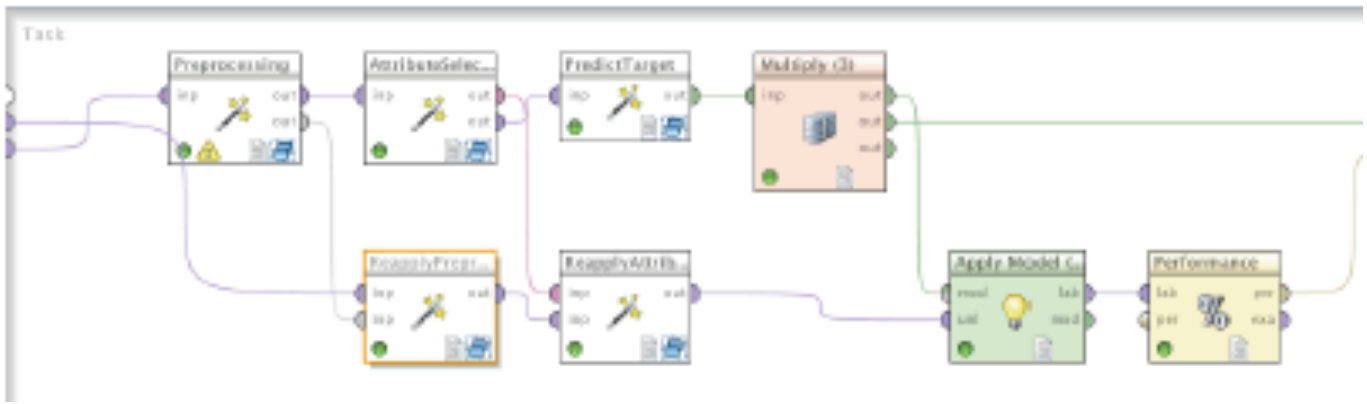
The aim is to generate various workflows / plans use meta-knowledge to select the best plan.

The Task / Method Decomposition

The selection is directed to the main goal (e.g. Predictive Modeling, Clustering etc.).

The top-level task is selected accordingly (e.g. Predictive Modeling with CV).

The nodes of the hierarchical planner are divided into
- completed nodes,
- nodes that still need to be planned (white nodes).

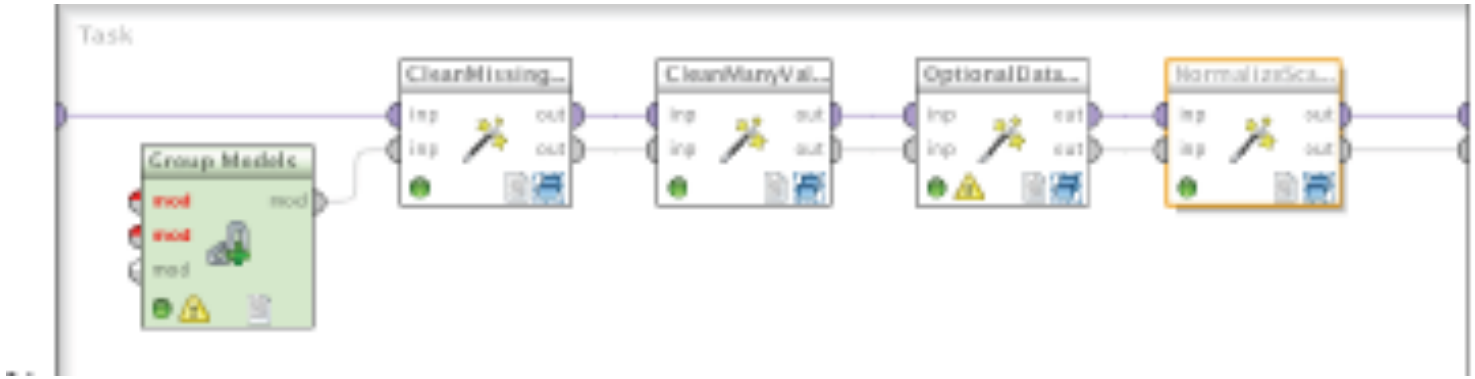


The Task / Method Decomposition

The system includes :

- Control / loop operators, like cross-validation.

- Operators for a group of operations for pre-processing attributes (the order of pre-preprocessing attributes in a group is irrelevant).



The system used task-methods decomposition grammar for hierarchical planning.

The Operator Models

The operators are structured in an inheritance hierarchy.

For instance, a basic classification operator inherits all the characteristics of the classification learner.

Each operators includes conditions and effects.

The description is formulated following specific formalism / language (e.g. concept expressions, Semantic Web Rule Language SWRL etc.)

The Operator Models

Example of a basic classification learner

”RM Support Vector Machine LibSVM C SVC linear”:
Equiv. class:
RM Operator and
(usesData exactly 1 DataTable) and
(producesPredictionModel exactly 1 PredictionModel) and
(simpleParameter kernel type value ”linear”) and
(simpleParameter svm type value ”minimal leaf size”) and
(operatorName exactly 1 {”support vector machine libsvm”})
Condition:
[MissingValueFreeDataTable and
(targetColumn exactly 1 CategorialColumn) and
(inputColumn min 1 Thing) and
(inputColumn only (ScalarColumn))
](?D)
→ RM Support Vector Machine LibSVM C SVC linear(?this),
simpleParameter svm type(?this,”C-SVC”),
simpleParameter kernel type(?this,”linear”)

The Planning System

The planner is implemented in FLORA2/XSB
(FLORA2/XSB is a knowledge base object oriented language.)

It employs a specialized Abox reasoner
that relies on external TBox.

Abox - assertional box (individual assertions, such as is-a),
Tbox – terminological box (properties e.g. all students are persons).

The system processes operators together with
their inputs, outputs, preconditions and effects
stored as OWL annotations.

The problem description includes:
- input description in terms of meta-data,
- goals / hints of the user.

Both are stored as a set of Abox assertions.

eIDA – programming interface to the reasoner / planner
used to plugin IDA to RapidMiner.

Using Rapid Miner IDA

eIDA – programming interface to the reasoner / planner used in plugin IDA to RapidMiner.

The RapidMiner IDA extension can be downloaded.

4. Metalearning for Model Selection in Other Domains

The approach described
that exploits metalearning and meta-level models
can be used in various other domains
for selecting the appropriate algorithm(s) or variant.

In the following we review the following domain:

- **parameter selection of classification algorithms,**
- **regression,**
- **time-series forecasting,**
- **sorting,**
- **constraint satisfaction,**
- **optimization.**

Metalearning for Parameter Selection

The approach that exploits metalearning & meta-level models can be used to select the appropriate parameters of classification algorithms.

Selection alternatives:
- **Appropriate type of kernel for SVM classifier (polynomial, radial, etc.)**
- **Appropriate kernel parameter (e.g. width of Gaussian kernel)**
- **etc.**

The process can be aided by the introduction of new specific features / data-characteristics.

Metalearning for Regression

Selection alternatives:
LDA, QDA, NN (‘00)

Specific features / data-characteristics:
Test of normality, Box’s M-statistics etc. (‘00)
Coefficient of variation, R^2 etc. (‘02)

Meta-learning model:
decision tree (‘00)
algorithm ranking model using k-NN (‘02)

Note:
00’ refers to one study carried out in 2000 (see K.Smith-Miles for details)

Metalearning for Time Series Forecasting

Selection alternatives:

Adaptive filtering, Holt's exponential smoothing, Winter's method ('97)

Simulated exponential smoothing, time-delay NN ('02)

Exponential Smoothing, ARIMA, Random walk, Backprop. NN ('05)

Specific features / data-characteristics:

number of turning points, autocorrelation coefficient, etc.('97)

Coefficient of variation, R^2 etc. ('02)

Meta-learning model:

Rule-based system ('00)

2-stage NN (determines group of algorithms and then algorithm) ('99)

Decision tree ('04)

Ranking model ('04)

Cluster Membership

Metalearning for Sorting

Selection alternatives:

- **InsertionSort, MergeSort, QuickSort** (‘01)
- **Five sorting algorithms** (‘03)

Specific features / data-characteristics:

- **Length of the sequence of integers** (‘01)
- **11 measures for the degree of presortedness** (‘03)
  - **(ex. number of inversions etc.)**

Meta-learning model:

- **Rule-based system** (‘01)
- **Decision tree, Naïve Bayes, Bayes network** (‘03)
- **More than 90% accuracy in selecting the fastest algorithm**

Metalearning for Constraint Satisfaction

The problem is characterized by a set of constraints

Selection alternatives:
- **3 solvers (‘04)**
- **7 high performance solvers** (‘07)
- **Selection of components of algorithms in real time** (‘07)

Specific features / data-characteristics:
- **48 features** (‘07)
- **78 features of QBF** (Quant. Boolean Formula) **instances**

Meta-learning model:
- **Linear regression to predict the logarithm of runtime** (‘01)
- **Empirical hardness model for each solver** (‘04)
  - **(phase transition from easy to hard problem)**

Dynamic algorithm selection (‘06, ‘07)

Metalearning for Optimization

Objective:
Find a solution to the problem that minimizes the cost (or maximizes the benefit)

Some optimization problems:
Travelling salesman problem (TSP)
Quadratic assignment problem (QAP)

Selection alternatives:
tabu-search, iterated local search, min-max ant system (‘04)

Specific features / data-characteristics:
4 features related to problem size + 4 measures of iter. search (‘04)

Meta-learning model:
Neural net to predict the performance gap and ranking (‘04)
Multi-label classification (‘11)
(J.Kanda et al.: Selection of algs. to solve TSP, ‘01)