Tutorial/Workshop @ ECML PKDD 2022

Meta-Knowledge Transfer/Communication in Different Systems

Jan N. van Rijn, Pavel Brazdil, Henry Gouk, Felix Mohr

Metalearning : Advanced Topics

Pavel Brazdil
Prof. Emeritus, U.Porto

23 Sept 2022

Overview

Duration ≈ 21 min.

1. Configuration Spaces (3-12) (Ch. 8)
   - 1.1 Adequacy of Configuration Spaces
   - 1.2. Reducing Configuration Spaces
2. Automating Data Science (13-20) (Ch. 14)
3. Designing Complex Applications (21-22) (Ch. 15)

1.1 Configuration Spaces (1)

Configuration space associated with workflow design

include all possible workflows (pipelines)
that can be constructed by combining
the given set of base-level elements using admissible operations

Typical base-level elements:

- Preprocessing operations and their hyperparameters
- ML algorithms (e.g. classifiers)
  and their hyperparameters (categorical, continuous, etc.)

1.1 Adequacy of Configuration Spaces (2)

Adequacy of Configuration Spaces wrt. Given Tasks

Informal analysis

If the configuration space is “too small”
the search might not yield good workflows
hence lead to suboptimal performance.

If the configuration space “too large”
the search might require too long
before the right workflow is found.

1.1 Adequacy of Configuration Spaces (3)

Let

S_C - **system** that has access to the **configuration space C**,

S_R - **system** that has access to the **required hypothetical configuration space R**,

T_C - **tasks** that can be solved adequately by S_C,

T_R – **future tasks** that should be solved by S_R.

The following situations can arise:

$R \equiv C$: we are well prepared for future tasks

$R \subset C$: future tasks can be solved, but C is unnecessarily large

Hence we may **apply reduction** to C

→ See the method further on

$C \subset R$: some future tasks cannot be solved,

Hence C needs to be extended

→ Methodology?

1.1 Adequacy of Configuration Spaces (4)

Who can set up the configuration space?

- **Designers** of the given metalearning/AutoML system (e.g., Auto-sklearn)
  For all purposes C is fixed
- **Experienced users (**specialists**)** of the given metalearning/AutoML system
  They have know-how toalter the system's options
- **Users** of the given metalearning/AutoML system
  They can edit a file with the set up
- **Other systems**
  Requires transfer/communication of definition of the configuration space

  Ex. AR* determines C' $\subset$ C, which is transferred to Auto-sklearn (future research)

1.2. Reducing Configuration Spaces (1)

Potential advantages:

- The **search** for the solution (model, predictions on a new task) is **facilitated** (takes less time)
- The **resulting model is "simpler"** - aids **explainability**

  It enables to identify useful:
  - Preprocessing methods
  - ML algorithms and their hyperparameters settings

References:
Brazdil, Soares & Pereira, 2001
Abdulrahman, S., Brazdil, et al., 2019
Brazdil et al., 2022, Book, Chapter 8
Hetlerović et al., 2022, Our workshop here

1.2. Reducing Configuration Spaces (2)

Method

used as preprocessing of workflows, a filter-like method:

- **Identify workflows** with **top performance** (e.g., accuracy)
  (eliminate sub-optimal workflows)
- **Process top performers to identify the non-redundant set**
  (eliminate the potentially redundant workflows)

1.2. Reducing Configuration Spaces (3)

Identify workflows with the **top performance:**

Initialize top performers ($\mathbf{a}_{Best}$) to an empty set.

Process all datasets $d_i \in D$:

- Identify all workflows that achieved top performance for d_i :

  Best performer plus all workflows **statistically equivalent** to best, **or**
  Best performer plus all workflows within the **top percentile of p%**

- Add the workflows identified to the top performers

Return the top performers

Note: It seem best to seek the top performers (experts) for each dataset (not globally best performers)!

1.2. Reducing Configuration Spaces (4)

Process top performers to **identify the non-redundant set**

Initialize non-redundant (reduced) set to empty set

Process all workflows in the set of top performers:

- If current workflow is non-redundant
  wrt. any element in the non-redundant set, add it to this set

Return non-redundant (reduced) set

Detecting whether two algorithm/workflows are **non-redundant**
≡ **do not exhibit a similar performance** (see next slide)

1.2. Reducing Configuration Spaces (4)

Detecting whether two algorithm/workflows are similar (or not)
(and hence one is potentially redundant):

The method is based on **performance comparison** of:

- **Aggregated** measure of performance on different datasets
  (e.g. measure $sim_{cos}(d_i, d_j)$)
- **Detailed** performance results **across all examples** on different datasets
  (e.g. Lee & Giraud-Carrier, 2011)

See “Metalearning for Algorithm/Workflow Selection”
Section 3 Using Dataset Characteristics

Visit our poster to see the details.

1.2. Reducing Configuration Spaces (5)

Some results

This methodology could be used to identify
a subset of 3 generally useful outlier elimination methods (OEMs)
from the initial set of 12 items.

Hetlerović et al., 2022,

See more details at the poster session!

2. Automating Data Science (DS) (1)

Ch. 14

Typical steps of DS process:

1. Defining the current problem/task
2. Identifying the appropriate domain-specific knowledge
3. Obtaining the data
4. Data preprocessing and other transformations
5. Automating model generation and its deployment
6. Automating report generation

So far, not explored much.
Future research should cover

Current focus in many articles

Objective of ML research
(not covered here)

2. Automating Data Science (2)

Step 1. Defining the current problem/task

- Problem understanding and **description**
  Typically elaborated by experts **in natural language**
- Determining the **task type** and **goals**
  E.g. classification of diseases (diagnoses)
- Generating **task descriptors**
  Automatic methods of **keyword extraction** could be exploited

2. Automating Data Science (3)

Step 2. Identifying the appropriate domain-specific data/models

Automatic methods of **determining the domain** can be used :

- by matching descriptors to metafeatures of a particular domain, or
- by using descriptors as features in classification (different domains represent the classes)

Step 3. Obtaining the data/models

- If data/models do not exist, plan how to obtain what is needed (as in
- Else determine where to obtain the data/models
- Resolve compatibility issues

2. Automating Data Science (4)

Step 4. Data transformations and preprocessing

- 4.1 Data transformation/**data wrangling**
- **4.2 Instance selection**/cleaning/outlier elimination
- **4.3 Data preprocessing**
  - Feature selection
  - Discretization
  - Nominal to binary transformation
  - Normalization/standardization
  - Missing value imputation
  - Feature generation (e.g., PCA or embeddings)

See the next slides

Discussed in many articles & books

2. Automating Data Science (5)

Step 4.1 Data wrangling

is a process of **transforming and mapping data**
from one data representation format (e.g., “raw” data) to another.

The **objective** is to **make it appropriate for subsequent processing**.
E.g., Convert information in a spreadsheet into a tabular dataset.

2. Automating Data Science (6)

Step 4.1 Data Wrangling - The Method

The **wrangling system exploits** often **both manual and ML methods**.

- The user shows how to transform a few examples.
- These are used as the **training examples** (input-output pairs) in the process of **learning** of **how to do the transformation** of the source data into the required format.
- Often it is necessary to learn to detect the data types (e.g. recognize dates)

As normally the number of training examples is small, traditional ML systems (e.g. rule-learning) are often used.

2. Automating Data Science (7)

Step 4.1 Some Data Wrangling Systems

- **Trifacta Wrangler** (Kandel et al., 2011)
  - Generates a list of possible transformations
- **System FOOFAH** (Jin et al., 2017)

  Searches the list of possible transformations to generate a program that carries out the required transformation.

  Typical task:

  Transform spreadsheet data into a relational format required by ML

  Some transformations:
  - Drop: deletes a column in the table
  - Move: relocates a column from one position to another
  - Merge, Split, Divide, Extract defined similarly

2. Automating Data Science (8)

Step 4.1 Some Data Wrangling Systems

- **Automated data transformation system** (Contreras-Ochando et al., 2019)

  It recognizes:

  - Type of the given named entity
    (date, email, name, phone number, etc.)
  - Country and the coding convention
    This activates the appropriate background knowledge (BK)
    This is useful for the recognition of the day “25” in “25-03-74”

- **SYNTH** (De Raedt et al., 2018)

  Can learn to carry out automatic completion of data
  in a set of worksheets

3. Designing Complex Applications (1)

Ch. 15

Two Research Lines/Systems:

Symbolic learning systems typically
Learn from: human-provided input,
examples / input–output pairs / sequences of steps
Exploit: symbolic representation (e.g., ILP), background knowledge
Applications: classical AI (incl. data wrangling)

Deep learning system (DNNs, CNNs etc.)
Learn from: large sets of examples (big data/data streams), embeddings
Exploit: sub-symbolic representation, pre-training + fine-tuning
Applications: vision, machine translation, logical implication

Are symbolic learning systems still useful?

3. Designing Complex Applications (2)

Why is the study of symbolic learning processes useful:

- Provides explainable solutions
- Humans use both symbolic and subsymbolic reasoning!
  Why did the symbolic reasoning arise?
- Permits to transfer/communicate the knowledge acquired to others
  (e.g. Newton's laws of motion)

Current/future research:

Systems should use **both symbolic** and **deep learning component** working in a symbiotic mode

How can we develop such systems?

Pascal Hitzler and K.Sarker, Neuro-symbolic AI, IOS Press, 2022
Pascal´s talk at this workshop!
Panel discussion

3. Designing Complex Applications (3)

Overview of some approaches in the area of symbolic learning

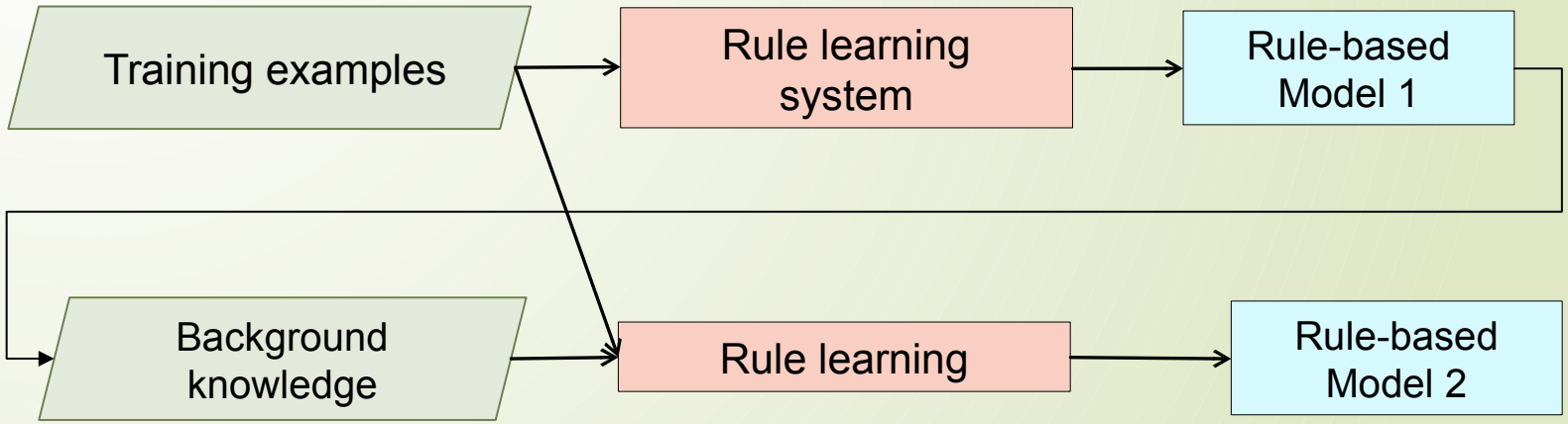
- Exploiting a richer set of operators
  - Conditional operators, Repeat operators, etc.
    (permitting advances in automatic programming)
- Changing the Granularity by Introducing New Concepts
  - Constructive induction, reformulation of theories (sets of rules), etc.
- Reusing New Concepts in Further Learning
  - Using acquired skills in learning more complex behaviour
- Iterative Learning
  - Revising rules by further learning
    (more details on the next slide)

3. Designing Complex Applications (3)

Iterative Learning – permits the revision of rules by further learning

The example discussed in Ch. 15 covers "insertion sort"

Jorge and Brazdil, 1996



It would be interesting if these techniques were reused in other settings (e.g. bioinformatics)

References

Abdulrahman, S., Brazdil, P., Zainon, W., and Alhassan, A. (2019). Simplifying the algorithm selection using reduction of rankings of classification algorithms. In ICSCA '19, Proc. of the 2019 8th Int. Conf. on Software and Computer Applications, Malaysia, p. 140–148.

Brazdil, P., Soares, C., and Pereira, R. (2001). Reducing rankings of classifiers by eliminating redundant cases. In Brazdil, P. and Jorge, A., eds, Proc. of the 10th Portuguese Conf. on Artificial Intelligence (EPIA2001). Springer.

PB Brazdil, JN van Rijn, C Soares, J Vanschoren, Metalearning: Applications to Automated Machine Learning and Data Mining (2nd ed.), Springer, 2022, Chapter 8

D Hetlerović, L Popelínský, P Brazdil, C Soares, F Freitas, On Usefulness of Outlier Elimination in Classification Tasks, International Symposium on Intelligent Data Analysis, 143-156, 2022

King, et al. (2009). The automation of science. Science, 324(5923):85–89.

Lee, J. W. and Giraud-Carrier, C. (2011). A metric for unsupervised metalearning. Intelligent Data Analysis, 15(6):827–841.

Steinruecken, C., et al. (2019). The Automatic Statistician. In Automated Machine Learning, Springer.