Towards the definition of learning systems with configurable operators and heuristics

Fernando Martínez-Plumed, Cèsar Ferri, José Hernández-Orallo, María José Ramírez-Quintana

NFMCP 2012

September 24, 2012
Towards the definition of learning systems with configurable operators and heuristics

Table of contents

1 Introduction
2 Setting
   - Principles
   - General Architecture
   - Rule and Program Repositories
   - System Operators
   - System Combiners
   - Reinforcement Module
3 Examples
   - Sequence Processing
   - Bunches of Keys
   - Web categorisation
4 Conclusions and Future Work
   - Conclusions
   - Future Work
Introduction

- Machine learning techniques dealing with structured data:
  - **Distances or kernel methods** can be applied to any kind of data (similarity functions).
  - **Inductive programming (ILP, IFP or IFLP)** are able to tackle any kind of data (first-order logic, term rewriting systems).
Introduction

The performance of these systems is linked to:

- a *transformation of the feature space* to a more convenient, flat, representation, which typically leads to incomprehensible patterns in terms of the transformed (hyper-)space
- use the original problem representation but *rely on specialised systems with embedded operators*

It is very difficult to have general systems which are able to deal with different kinds of complex data.
Introduction

We present a general rule-based learning setting where operators can be defined and customised for each kind of problem.

- The generalisation operator to use depends on the structure of the data.
- Adaptive and flexible rethinking of heuristics, with a model-based reinforcement learning approach.
Setting

- **Machine learning operators** are the tools to explore the hypothesis search space.
  - Some operators are usually associated to some heuristic strategies (e.g., generalisation operators and bottom-up strategies).
- Operators can be modified and finetuned for each problem:
  - Different to the use of feature transformations or specific background knowledge.
- This is a challenging proposal not sufficiently explored in machine learning.
Operators can be written or modified by the user

- We need a language for defining operators which can integrate the representation of:
  - Examples.
  - Patterns.
  - Operators.
We have chosen a powerful popular programming language, **Erlang**:

- A functional programming language, with **reflection** and **higher-order primitives**.
- Operators can be properly linked with the data structures used in the examples and background knowledge, so making the specification of new operators easier.
- The language also sets the general representation of examples as equations, patterns as rules and models as sets of rules.
Towards the definition of learning systems with configurable operators and heuristics

General Architecture

Figure: Prototype System Architecture
Two internal repositories containing rules and programs.

Initially, the set of rules $R$ is populated with the positive evidence $E^+$ and the set of programs $P$ is populated defining unitary programs from the rules of $R$.

Both repositories are updated at each step of the algorithm:

1. The Rule Generator builds new rules ($r^{new}$) and they are added to $R$.
2. By applying the combiners, ($r^{new}$) is mixed with the programs in $P$ generating a new program $p^{new}$, and it is added to $P$. 
The user can define his/her own set of operators, especially suited for the data structures of the problem: Adaptive system.

An operator is defined as a function which is applied to a rule in order to generate new rules:

- Given a rule $f(X) \rightarrow Y$ where the input attribute $X$ is a list, the operator can extract the head of $X$ and return it as the rhs of the new rule.
- The operator could be defined as:

$$\text{takeHead}(f(X) \rightarrow Y) \ [\text{when } X \text{ is a List}] \rightarrow (f(X) \rightarrow \text{head}(X))$$
Combiners evolve the population of programs.

- **Addition**: adds the program that results from joining the new rule $r_{new}$ generated by the Rule Generator with the best program (in terms of optimality);

- **Union**: joins the two best programs (also in terms of optimality) in $P$. 
Towards the definition of learning systems with configurable operators and heuristics

Reinforcement Module

- A reinforcement learning module guides the *Rule Generator* in each step of the algorithm.
  - $S$ represents the system state as the set composed by $R$ and $P$.
  - An action $A$ is a tuple $< r_i, o_i >$ where $r_i$ is a rule and $o_i$ is an operator.
- Given an state $S$, an action $A$ is chosen by the *Heuristic Model* and sent to the *Rule Generator*. This creates new rules (and programs), which causes the system to move to a new state.
Initially, the *Heuristic Model* does not have enough evidence and the choice is random, but after a few iterations, the model is learnt by using a machine learning technique.

This model is trained to predict the reward after a given action $A$, and with it we choose the action which maximises the estimated reward.

**Rewards:**

- From the optimality $Opt^{new}$ of the new program $p^{new}$, the *Reinforcement Module* calculates a reward $Rew$.
- $Rew$ is used to update the optimality of the action $A = \langle r_{i}, o_{i} \rangle$. 
Towards the definition of learning systems with configurable operators and heuristics

Examples

Sequence Processing

- Learning a transformation over the words formed by a given alphabet.
  - Alphabet $\Sigma = \{a, t, c, g, u\}$
  - Transformation just replaces $t$ with $u$.

<table>
<thead>
<tr>
<th>Instance</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{trans}([t, c, g, a, t]) \rightarrow [u, c, g, a, u]$</td>
</tr>
</tbody>
</table>
### Examples

**Sequence Processing**

#### Background Knowledge

\[ f_{at}(a) \rightarrow t; \ f_{cg}(c) \rightarrow g; \ldots \]  

(1)

#### Operators

\[ \text{applyMap}(\text{trans}(X) \rightarrow Y) \Rightarrow \text{trans}(X) \rightarrow \text{map}(V_F, X) \]  

(2)

\[ \text{addBK}_f(\text{trans}(X) \rightarrow \text{map}(V_F, X)) \Rightarrow \text{trans}(X) \rightarrow \text{map}(f, X) \]

(3)

\[ \text{genPat}(\text{trans}(X) \rightarrow Y) \Rightarrow \text{trans}(V_S) \rightarrow Y \]
Towards the definition of learning systems with configurable operators and heuristics

Examples

Sequence Processing

There is a simple sequence of operator applications which turns a simple example into a general solution.

Given the instance \( \text{trans}([t, c, g, a, t]) \rightarrow [u, c, g, a, u] \):

Solution **Sequence Processing** problem

\[
\text{genPat}(\text{trans}([t, c, g, a, t]) \rightarrow [u, c, g, a, u]) \quad \Rightarrow \quad \text{trans}(V_S) \rightarrow [u, c, g, a, u] \\
\text{applyMap}(\text{trans}(V_S) \rightarrow [u, c, g, a, u]) \quad \Rightarrow \quad \text{trans}(V_S) \rightarrow \text{map}(V_F, V_S) \\
\text{addBK}_{f_{tu}}(\text{trans}(V_S) \rightarrow \text{map}(V_F, V_S)) \quad \Rightarrow \quad \text{trans}(V_S) \rightarrow \text{map}(f_{tu}, V_S)
\]
Consider the well-known problem of determining whether a key in a bunch of keys can open a door.

Each instance is given by a bunch of keys, where each key has several features: two-level structure (sets of lists).

\[
\text{opens}\left(\left[\left[\text{abloy, 3, medium, narrow}\right], \left[\text{chubb, 6, medium, normal}\right]\right]\right) = \top
\]
Towards the definition of learning systems with configurable operators and heuristics

Examples

Bunches of Keys

Bunches of Keys

Background Knowledge

\[ \text{setExists}(\text{Key}, \text{Bunch}) \] (4)

Operators

\[ \text{addBK}(\text{opens}(X) = \top) \Rightarrow \text{opens}(X) \rightarrow \text{setExists}([], X) \] (5)

\[ \text{KCond}_{\text{cond}_i}(\text{opens}(X) \rightarrow \text{setExists}(C, X)) \Rightarrow \]
\[ \text{opens}(X) \rightarrow \text{setExists}([\text{cond}_i | C], X) \] (6)

\[ \text{genPat}(\text{opens}(X) = Y) \Rightarrow \text{opens}(V_L) \rightarrow Y \] (7)
Bunches of Keys

- If the prototype and operators are provided, given the original evidence for this example (five $\top$ instances and four $\bot$ instances), it will return the following definition:

  Solution *Key of Bunches* problem

  \[
  \text{opens}(X) \rightarrow \text{setExists}([\text{abloy}, \text{medium}], X)
  \]

- A *bunch of keys opens the door if and only if it contains an abloy key of medium length.*
Web categorisation

- **Web classification problem**: web pages are assigned to pre-defined categories mainly according to their content (content mining).

- The evidence of the problem is modelled with 3 parameters described as follows:
  - **Structure**: the graph of links between pages is represented as ordered pairs where each node encodes a linked page.
  - **Content**: the content of the web page is represented as a set of attributes with the keywords, the title, etc.
  - **Use**: the information derived from connections to a web server which is encoded by means of a numerical attribute with the daily number of connections.
Web categorisation

- The goal of the problem is to categorise which web pages are about sports.
- A training example may look like this:

\[
sportsWeb(Structure, Content, Connections) \rightarrow \top
\]

where:

- **Structure** =
  \[
  \{[\text{olympics, games}], [\text{swim}]\}, \{[\text{swim}, \text{win}]\}, \{[\text{win}, \text{medal}]\}\]
- **Content** = \[
  \{\text{olympics, 30}\}, \{\text{held, 10}\}, \{\text{summer, 40}\}\]
- **Connections** = 20
Web categorisation

Background Knowledge

\[
\text{graphExists}(\text{Edge, Graph})
\]  \hspace{1cm} (8)

\[
\text{setExists}(\text{Key, List})
\]  \hspace{1cm} (9)

Operators

\[
\text{addBK}_{\text{graph}}(\text{sportsWeb}(S, C, U) \rightarrow \top) \Rightarrow
\]

\[
\text{sportsWeb}(S, C, U) \rightarrow \text{graphExists}([[], []], S)
\]  \hspace{1cm} (10)

\[
\text{linkl}_{\text{cond}, i}(\text{sportsWeb}(S, C, U) \rightarrow \text{graphExists}([X, Y], S)) \Rightarrow
\]

\[
\text{sportsWeb}(S, C, U) \rightarrow \text{graphExists}([[\text{cond}, i|X], Y], S)
\]  \hspace{1cm} (11)
Towards the definition of learning systems with configurable operators and heuristics

Web categorisation

Operators

\[
\begin{align*}
\text{linkr}_{\text{cond}_1}(\text{sportsWeb}(S, C, U) \rightarrow \text{graphExists}([X, Y], S)) & \Rightarrow (12) \\
\text{sportsWeb}(S, C, U) \rightarrow \text{graphExists}([X, [\text{cond}_1|Y]], S) \\
\text{genPat}_1(\text{sportsWeb}(S, C, U) \rightarrow \top) & \Rightarrow (13) \\
\text{sportsWeb}(V_S, C, U) \rightarrow \top
\end{align*}
\]

There are also some other operators to generalise the second and third arguments.
Towards the definition of learning systems with configurable operators and heuristics

Web categorisation

- Our system found the following program which defines the *sportsWeb* function:

Solution *Key of Bunches* problem

\[
\begin{align*}
\{ \text{sportsWeb}(V_S, V_C, V_U) \} & \rightarrow \text{graphExists}([\text{final}, \text{match}], V_S). \\
\text{sportsWeb}(V_S, V_C, V_U) & \rightarrow \text{setExists}([\{\text{athens}\}], V_C). \\
\text{sportsWeb}(V_S, V_C, V_U) & \rightarrow \text{setExists}([\{\text{europe}\}], V_C). \\
\end{align*}
\]

- If the word ‘athens’ or ‘europe’ appears in Content, and Structure contains the link \{[final], [match]\} then this is a sport web page.
Conclusions

- More general systems can be constructed by a flexible operator redefinition and the reuse of heuristics across problems and systems.
- In order to reduce the search space we rely on the definition of customised operators, depending on the data structures and problem at hand.
- We need a language for expressing operators for defining new operators easily.
Conclusions

- The use of different operators precludes the system to use specialised heuristics for each of them.
- We have proposed this as a decision process, where operators are actions to be taken, and this is also seen as a reinforcement learning problem.
Future Work

- Transforming the prototype into a learning system, including all the issues in the architecture.
- We need to further develop and refine the heuristics module of the system:
  - Improved description of the state
  - Better reinforcement learning models (which could eliminate many useless explorations of the search space).