Symbolic AI: Task 2

Johannes Fürnkranz, Van Quoc Phuong Huynh, Josef Küng

26.04.2021

The aim of this task is to help students get acquainted with Inductive Logic Programming in general, and with a particular implementation called Aleph. Aleph is an advanced ILP system with many different options. A detailed handbook can be found at `http://www.cs.ox.ac.uk/activities/programinduction/Aleph/`.

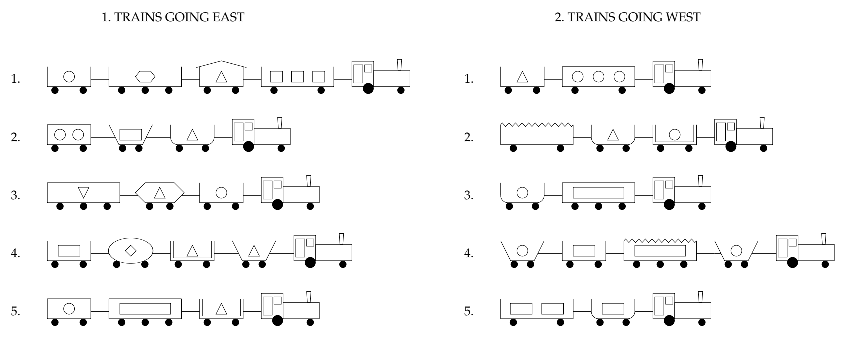
1. *Preparation:*

   (a) Download and install Aleph. In Moodle, we provide you with a slightly modified version 'aleph_swi.pl' that should work with recent versions of SWI-Prolog.[1]

   (b) Take a look a the manual at `http://www.cs.ox.ac.uk/activities/programinduction/Aleph/` and familiarize yourself with the basic workings of the system.

   (c) You can find a set of simple examples at `http://www.comlab.ox.ac.uk/oucl/research/areas/machlearn/Aleph/misc/examples.zip`. Download and save them, you will need them for the subsequent tasks.

2. *Getting Started:* Machine Learning pioneer Ryszard Michalski has come up with the following by now classical example for a task that requires first-order logic:



   The task is to find a rule that discriminates eastbound trains from westbound trains, based on the size and type of cars (number of wheels) and what loads they carry. As trains have different numbers of cars, it is hard to represent this in a conventional, propositional learning setting.

[1]Note that Aleph is also available as a SWI-Prolog package, which can be installed via `pack_install(aleph)` and activated via `use_module(library(aleph))`. This uses a slightly different interface (cf. `https://www.swi-prolog.org/pack/file_details/aleph/prolog/aleph.pl`). However, we could not get it to work. Let us know if you have more luck...

You can find this example encoded for Aleph in the `examples/trains` directory. Load Aleph (`consult('aleph-swi')`), change into the example directory (e.g., `chdir(examples/trains)`), and load all files with `read_all(train)`. You will probably get quite a few warnings, which you can ignore (but you should not get any errors).

(a) Look at how the positive and negative examples and, in particular, the background knowledge is encoded in the files `train.[f,n,b]`. Try to find a solution to the problem by hand.

(b) Use Aleph to learn a solution by calling `induce`. You can now look at the learned rule(s) with `show(theory)`, at the examples with `show(pos)` and `show(neg)`. Familiarize yourself with other options of `show/1` as well.

Inspect the learned rule, interpret it, and compare it to what you have got in (a).

(c) Look at the bottom clause (`show(bottom)`), try to interpret it, and show that the learned rule subsumes the bottom clause.

3. *Recursion:* In the folder `examples/recursion` you can find files that allow you to learn a definition for `member/2`.

(a) Use them to learn the `member/2` predicate. Also, look at the bottom clause. Why do you think it is so simple?

(b) * Define background knowledge, training and test files for learning at least one of the following recursive list processing programs: `last([b, a, c], c)`, `max([4, 2, 6], 6)`, `length([a, b, c], 3)`, `element_at(2, [a, b, c], b)`.

(c) ** Try to learn a more complex predicate. You can think of recursive definitions of sorting, the Towers of Hanoi problem, etc.

4. *Generalization**: Your work in this exercise is to use Aleph to learn rules from a training dataset and to test the rules with a testing dataset. The dataset files, 'training_dataset' and 'testing_dataset' are available in Moodle. The data sets are originally from 'German Credit Data' data set at `https://archive.ics.uci.edu/ml/datasets/statlog+(german+credit+data)` that were prepared for you such as removing some numeric attributes, train-test splitting. You can use the link for more detail information about attributes and values.

(a) Prepare the background knowledge, training, and testing files to be processed by Aleph. You certainly need to write a short program in any other programming languages e.g. Python to prepare the files because the number of examples is big enough that you maybe cannot edit the files only by hand. After all files are already, let Aleph to learn a theory and to test it. Report the error rate or accuracy and run time as well on the test set. (*Hint*: In Moodle, you can find two very small data sets 'sample_training_data' and 'sample_testing_data' that you can prepare files by hand to be processed by Alepth. Then using the files as templates for the bigger datasets. For rule evaluation, you have to test twice, once on the positive and once on the negative examples).

(b) Try to optimize the prediction performance (get a lower error rate) by playing around with some of Aleph's parameters (such as `evalfn`, `noise`, `search`, etc.). Report the error rates on the training and the test sets for at least 5 configurations that yield different results, as well as the respective numbers of rules and/or conditions. Also report if some (other) configurations failed.

5. *Try something interesting**: You can choose one of the following two options:

    (a) Finding your own an interesting application of ILP and try to solve it, some ideas such as something related to your first task, something related to knowledge bases and the Semantic Web which will follow later in the course, etc.

    (b) From the link `https://www.doc.ic.ac.uk/~shm/applications.html` you can find some ILP applications and the respective data sets. Each group choses one to solve and registers your choice on the shared worksheet so that no two groups do the same one.

6. *Submission*: You only report your solution for exercises marked with "*" or "**". In that, exercises with "*" are mandatory and ones with "**" are bonus, not mandatory.

    Send your Task 2 result to vqphuynh@faw.jku.at until **June 8th, 2021, 23h:59'**. Any submissions after the deadline will be penalized 20% from the total mark of this task. All files must be compressed in a .zip file with the file name in format "group_xx", e.g. "group_01". The package includes:

    (a) All files to be processed by Aleph.

    (b) A slide file for your presentation on the only date 15.06.2021 at 15:15

    (c) A report file for your solutions and experiments.

Note that this may turn out to be harder than you think, so try to aim for a simple task. Try hard to succeed, but it is also o.k. to provide a report on a failure (why didn't it work as expected).