SIMULA - ML Course – 24 Aug. 2016 –
An AI Perspective on Symbolic Machine Learning

Arnaud Gotlieb

Machine Learning is a vast research domain which can be approached through many different angles. This course will follow an Artificial Intelligence perspective to enter the discipline by following one its promising historical development line. In this context Learning can be seen as the acquisition of explicitly represented domain knowledge, through the construction and modification of expressions in a formal language, such as Logical formulae or relations (Luger 05). In a first part, the course will review some basic knowledge, methods and algorithms used in symbolic machine learning, including:

- ID3 Decision Tree Induction algorithm (Quinlan 86)
- Version Space Search and Candidate Elimination algorithm (Mitchell 82)
- Constraint Acquisition (Bessiere 15)

In a second part, the course will be taken in the Lab, with practical exercises in PROLOG (Clocksin 03). A gentle introduction to the language will be given, and then different levels of exercises will be proposed to the participants depending on their level of practice of the programming language.

No prerequisite are necessary to follow the course, apart from positive curiosity and thirst to learn.

References:

(Luger 05) G. F. Luger - Artificial Intelligence: Structures and Strategies for Complex Problem Solving – Fifth edition – Addison Wesley – 2005

(Mitchell 82) T.M. Mitchell – Generalization as search. Artificial Intelligence 18(2):203-226, 1982

(Quinlan 86) J.R. Quinlan – Induction of decision trees. Machine Learning, 1(1):81-106, 1986

(Bessiere15) C. Bessiere, F. Koriche, N. Lazaar, B. O'Sullivan – Constraint Acquisition – Artificial Intelligence – In Press

(Clocksin 03) W. F. Clocksin, C. S. Mellish – Programming In Prolog – 5th edition – Springer, 2003

Instructions to install SICStus 4.3

```
To download SICStus Prolog release 4.3,
please visit the following URL:

       http://www.sics.se/sicstus/products4/
```

The login name is "sicstus4" and the password is "H75umirT".

```
SICStus Prolog is distributed as a self-extracting executable
(Windows) resp. a gzipped tar-file (UNIX).
It requires a decryption key to install; see below.
```

***** Your license information for download 'darwin' is as follows:**

```
Decryption key:     mugeujvbbuwm
Site name:          SP4.3simulaCourse
License code:       78g3-bft2-nzg2-bd5n-gusv
Expiration date:    20160911
```

This information is entered either during installation or by modifying your license information after installation. This is done by typing the following at a command prompt:

```
splm -i SP4.3simulaCourse
splm -a sicstus4.3_darwin 20160911 78g3-bft2-nzg2-bd5n-gusv
```

***** Your license information for download 'linux' is as follows:**

```
Decryption key:     bmgghubktwqz
Site name:          SP4.3simulaCourse
License code:       bc5d-a5f4-9f76-bf8f-ddo7
Expiration date:    20160911
```

This information is entered either during installation or by modifying your license information after installation. This is done by typing the following at a command prompt:

```
splm -i SP4.3simulaCourse
splm -a sicstus4.3_linux 20160911 bc5d-a5f4-9f76-bf8f-ddo7
```

***** Your license information for download 'win32' is as follows:**

```
Decryption key:     imgmepruwavp
Site name:          SP4.3simulaCourse
License code:       bebc-af6u-ru4r-csz2-fmh2
Expiration date:    20160911
```

This information is entered either during installation or by modifying your license information after installation. This is done from the 'Settings' menu item in the SICStus application (spwin.exe).

By ordering, you confirm that you have read and agreed to the terms of the SICStus Prolog 4 license agreement applicable for your license; see:
http://www.sics.se/sicstus/license4-eval.html

Lab Work – Symbolic Machine Learning

Date: 24 Aug. 2016

Prerequisites:

- SICStus Prolog version 4.3.3 (https://sicstus.sics.se/download4.html)
- Availability of a text editor

Notes: The exercises are labelled with 3 distinct levels of difficulty, namely,
* - *very simple*, ** - *simple*, *** - *less simple.*
Each topic section contains exercises with different labels.

For those who are already familiar with the Prolog language (who have a development experience in Prolog), we suggest to go ahead with only the ***-labeled exercises.
For those who are new to the language, but want to learn Prolog in depth, we suggest to proceed with the **-labeled exercises, after having checked that they master the *-labeled exercises.
For those who just want to get introduced to Prolog (without necessarily understanding the language in depth), we suggest to proceed only with *-labeled exercises.

A. Atoms, Variables and Terms, Clauses and Predicates

socrates.pl	**arcs.pl**

man(socrates).
man(plato).
 	
woman(judith).
 	
mortal(X) :- man(X).
mortal(X) :- woman(X).
	arc(a, b).
	arc(b, c).
	arc(b, d).
	arc(c, d).
	arc(d, e).
	
	path(X, X, [X]).
	path(X, Y, [X|Path]) :-
	  arc(X, Z),
	  path(Z, Y, Path).

(- very simple)*
By interpreting the Prolog program **socrates.pl**, and querying the program with the following requests, predict the expected result and identify the atoms, variables, terms and clauses and predicates in this Prolog program.

```
?- mortal(plato).
?- mortal(judith).
?- mortal(omer).
```

```
?- mortal(ANY).
```

*(** - simple)*

By interpreting the Prolog program **`arcs.pl`**, and querying the program with the following requests, predict the expected result:

```
?- path(a, e, Path).
?- path(e, a, Path).
?- path(b, d, Path).
?- path(b, Y, Path).
?- path(X, d, Path).
```

*(*** - less simple)*

Considering **`arcs.pl`**, how will you enumerate all paths from `a` to `e` (which query will you type in)?

B. Terms Unification

Considering the following requests, can you predict and write down Prolog answers? In any case, type-in and check the result.

(- very simple)*

```
?- t(X, b, Z) = t(a, Y, 4).
?- t(X, b, Z) = t(a, X, 4).
?- t(X, 1, p(Y, 3)) = t(a, Z, T).
```

*(** - simple)*

```
?- t(X, 1, p(X, 3)) = t(a, Y, Z).
?- t(X, 1, p(X, 3)) = t(a, Y, p(Y, Z)).
?- t(X, T, p(X, 3)) = t(a, Y, Y).
```

*(*** - less simple)*

```
?- X = p(X).
?- t(X, t(Y)) = t(p(Y), X).
?- t(X, t(Y)) = t(t(Y), X).
?- t(X, t(Y)) = t(t(Y), X), X = Y.
```

C. Bactracking

A **Latin square** is an $n \times n$ board filled with n different symbols, each occurring exactly once in each row and each column. Here is an example of a 3x3 latin square:

1	2	3
3	1	2
2	3	1

(- very simple)*
Using backtracking, write a program which successfully gives the value 1, 2 and 3 to a variable X? How to use it to populate a 3x3 latin-square?

*(** - simple)*
Write a program which creates all 3x3 latin-squares through backtracking. How many distinct 3x3 latin-squares can be generated?

*(*** - less simple)*
Write a program which creates all NxN latin-squares through backtracking.

D. Lists Manipulation

(- very simple)*
Given a list of N integers, write a Prolog program which test if a given integer belongs or not to that list.

*(** - simple)*
Given two lists of atoms, write a Prolog program which concatenate these two lists.

*(*** - less simple)*
Given a list of N integers, write a sorting program.

E. Non-deterministic Search

Arcs2.pl

```
arc(a, b).
arc(b, c).
arc(b, d).
arc(c, d).
```

```
arc(d, a).
arc(d, e).
arc(e, a).

path(X, X, [X]).
path(X, Y, [X|Path]) :-
   arc(X, Z),
   path(Z, Y, Path).
```

(- very simple)*

Using the `write/1` predicate, write a program which enumerates all triplets (X, Y, Z) where X, Y, Z takes any value between 1 and 3 and write them on the console.

*(** - simple)*

Write a program which enumerates all paths of the `arcs2.pl` program starting from the vertex `a`, and getting back to a. How many are they? Implement a strategy to cope with the problem by limiting the length of the paths. How many paths are there with length less than 10?

*(*** - less simple)*

Write a program which searches for the path starting from the vertex `a` in the `arcs2.pl` program, and getting back to a, which has the largest number of occurrences of a, and of length less than 50.

F. Anti-Unification

Considering the following terms, can you compute their *least general unifier*? In any case, type-in and check the result.

(- very simple)*

`t(a, b, 1)` and `t(a, c, 3)`
`t(t(1), b, p(a,3))` and `t(t(2), a, p(b,3))`

*(** - simple)*

`t(X, 1, p(X, 3))` and `t(a, 3, p(Y, 3))`
`t(X, p(b, 3))` and `t(a, p(X, 3))`
`t(X, p(a,3), p(a, 3))` and `t(a, Y, Y)`

Verify the results by using the `term_subsumer(+t1, +t2, -t)` predicate of library(terms).

*(*** - less simple)*

`X` and `p(X)`
`t(X, t(X))` and `t(t(X), X)`
`t(t(X), X)` and `t(t(X), X)`

Verify the results by using the `term_subsumer(+t1, +t2, -t)` predicate of library(terms).

G. Decision Tree Induction

Weather forecast with the following attributes:
outlook, with values {sunny, overcast, rain}
temperature, with values {cool, mild, hot}
humidity, with values {high, normal}
windy, with values { true, false}

Let suppose that we have only two values for classifying a set of data: either **P(positive)** or **N(negative).** For example, P means that the combination of attributes is possible (it has already been observed in the past), while N means that the combination is probably not possible (it has never been observed in the past).

Let S be the following training set:

1	sunny	hot	high	false	N
2	sunny	hot	high	true	N
3	overcast	hot	high	false	P
4	rain	mild	high	false	P
5	rain	cool	normal	false	P
6	rain	cool	normal	true	N
7	overcast	cool	normal	true	P
8	sunny	mild	high	false	N
9	sunny	cool	normal	false	P
10	rain	mild	normal	false	P
11	sunny	mild	normal	true	P
12	overcast	mild	high	true	P
13	overcast	hot	normal	false	P
14	rain	mild	high	true	N

(- very simple)*
Considering the training set S given above (example of weather forecast), write several Prolog terms representing distinct decision trees associated to S

*(** - simple)*
Write a Prolog program which creates one decision tree from the training set S given above (example of weather forecast), by using a simple attributes ordering.

*(*** - less simple)*
Write a Prolog program which creates all possible decision trees from the training set given above (example of weather forecast) and which selects a simplest one by considering the depth of the tree.