1 LP^{MLN} Learning System Usage

This document provides information needed for install and use the LP^{MLN} learning system.

The LP^{MLN} learning system includes the following components:

- Gradient Descent Based Learning with MC-ASP
- Pseudo-likelihood Based Learning
- Gradient Descent Based Learning with Gibbs Sampling
- Gradient Descent Based Learning with Metropolis-Hasting Sampling
- Sampling Based Inference

1.1 Installation

The system requires the following software package:

- Python 2.7.x
- clingo 5.x (The system assumes clingo 5 can be executed with command "`clingo5`")
- sympy
- Boost

To install the system, follow the steps below:

1. Unzip the compressed file

   ```
   unzip lpmln-learn.zip
   ```

2. Go to `code` directory

   ```
   cd lpmln-learn/code
   ```

3. Compile `lpmlncompiler` with the command

   ```
   g++ -g -std=c++11 lpmlncompiler.cpp -o lpmlncompiler -L /usr/lib -lboost_regex
   ```

4. Compile `negConv` with the command

   ```
   g++ -g -std=c++11 negConv -o negConv -L /usr/lib -lboost_regex
   ```

5. Compile `clingo3to4` with the command

   ```
   cd Clingo3to4
   make
   cp clingo3to4 ../
   ```

1.2 Input Format

For learning tasks, the input program is a program of the input format of LPMLN2ASP. The weights to be learned needs to be represented with the placeholder `@getWeight(idx)`, where `idx` is the id of the rule (that LPMLN2ASP generates for the rule). For example,

```
#domain person(X).
#domain person(Y).
person(0..9).

@getWeight(1) pf1(X).
cancer(X) :- smokes(X), pf1(X).
@getWeight(2) pf2(X, Y) :- friends(X, Y).
smokes(Y) :- friends(X, Y), smokes(X), pf2(X, Y).
```

Note that the domains of variables that occur in the rules whose weights are to be learned need to be specified with `#domain` statement.

For learning with MC-ASP sampling, the evidence file is simply a list of constraints, specifying truth value of atoms, for example:

```
:- not smokes("edward").
:- not smokes("frank").
:- not smokes("gary").
:- smokes("bob").
:- smokes("chris").
:- smokes("daniel").
:- smokes("helen").

:- not cancer("anna").
:- not cancer("edward").
:- cancer("bob").
:- cancer("frank").
:- cancer("chris").
:- cancer("daniel").
:- cancer("gary").
:- cancer("helen").
```

For learning with pseudolikelihood, Metropolis-Hasting sampling and Gibbs sampling, the evidence file should specify the truth value of all atoms, of the form

```
atom_name argument1;...;argumentN truth_value
```

`truth_value` is 0 (for FALSE) or 1 (for TRUE).

An example is

```
p 0;0 0
p 0;1 1
```

```
q 0;0 0
q 0;1 1
```

which means $p(0,0)$ and $q(0,0)$ are false, $p(0,1)$ and $q(0,1)$ are true.

For sampling based inference, a file specifying the domains of variables is required. The file should be of the form

```
predicate1 comb1_arg1,...,comb1_argN&...&combM_arg1,...,comb1_argN
...
```

For example,

```
pf1 "t";0&"f";0&"t";1&"f";1
pf2 "t";0&"f";0
```

specifies that the first argument of `pf1` and `pf2` is "`t`" or "`f`", the second argument of `pf1` ranges over $\{0,1\}$ and the second argument of `pf2` can only be `0`.

1.3 Gradient Descent Based Learning with MC-ASP

Learning parameters such as learning rate and maximum number of iterations can be configured in the file `code/learn-mcsat.py` and `code/learn-mcsat-em.py`.

1.3.1 With Complete Evidence

The command line to run gradient descent based learning with MC-ASP is

```
python code/learn-mcsat.py path/to/program path/to/evidence
```

For example

```
python code/learn-mcsat.py benchmarks/smoke/smoke.lpmln
    benchmarks/smoke/smoke-evidence.txt
```

will run MC-ASP based learning on the smoke example.

1.3.2 With Incomplete Evidence

When the evidence is incomplete, the command line is

```
python code/learn-mcsat-em.py path/to/program path/to/evidence
```

For example

```
python code/learn-mcsat-em.py benchmarks/smoke/smoke_plg.lpmln
     benchmarks/smoke/smoke-evidence.txt
```

will run MC-ASP based learning on the smoke example with pf atom.

1.4 Pseudo-likelihood Based Learning

Open the file `lpmln-learn.py`. At the beginning of the file, set the value of the variable `learning_alg` to be "`code/learn-pseudolikelihood.py`".

The command to execute pseudo-likelihood based learning is

```
python lpmln-learn.py path/to/program path/to/evidence
```

1.5 Gradient Descent Based Learning with Gibbs Sampling and Metropolis-Hasting Sampling

Open the file `lpmln-learn.py`. At the beginning of the file, set the value of the variable `learning_alg` to be "`code/learn-mhsampling.py`" for Metropolis-Hasting sampling or "`code/learn-gibbssampling.py`" for Gibbs sampling.

The command to execute gradient descent based learning with Metropolis-Hasting sampling or Gibbs sampling is

```
python lpmln-learn.py path/to/program path/to/evidence
```

1.6 Sampling Based Inference

To execute Metropolis-Hasting sampling based inference, the input program needs to be turned into an ASP program first, then marginal inference can be executed with

```
clingo code/marginal-mhsampling.py program_turned_into_ASP
```

for metropolis-hasting sampling based marginal inference or

```
python code/marginal-mcsat.py program_turned_into_ASP query_atom domain_file
```

for MC-ASP sampling based marginal inference.