GSoC 2018 - Markov Logic Networks in Python: pracmln

25.03.2018

Kaivalya Rawal - BITS Pilani, Goa Campus

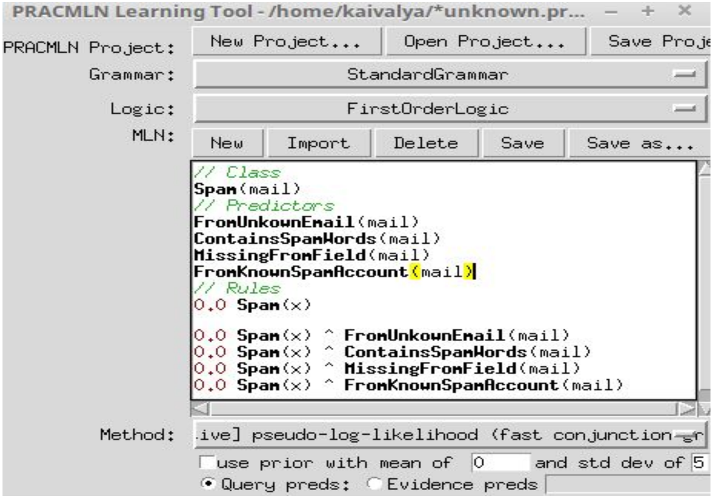
[redacted]@gmail.com | github.com/kaivalyar | linkedin.com/in/kaivalyar | +91 [redacted]

Assignment

I have completed the task assigned for this project, and sent my solution to Daniel Nyga at [redacted] (on 24th March 2018), containing relevant documents, the pracmln project file, and a brief explanation of my work.

Overview

Markov Logic Networks (MLNs) are a generalization of both first order logic and probabilistic graphical models, thus forming a powerful mechanism of uncertain inference. They were first introduced by Pedro Domingos and Matt Richardson, and the associated Alchemy software suite can be used to perform statistical relational learning (SRL) and probabilistic logic inference based on the Markov logic representation. Another implementation of MLNs can be found in *pracmln*, an open-source toolkit for SRL from the Institute for Artificial Intelligence at the University of Bremen.



PRACMLN Project:

Grammar: StandardGrammar

Logic: FirstOrderLogic

MLN:

```
// Class
Spam(mail)
// Predictors
FromUnkownEmail(mail)
ContainsSpamWords(mail)
MissingFromField(mail)
FromKnownSpamAccount(mail)
// Rules
0.0 Spam(x)

0.0 Spam(x) ^ FromUnkownEmail(mail)
0.0 Spam(x) ^ ContainsSpamWords(mail)
0.0 Spam(x) ^ MissingFromField(mail)
0.0 Spam(x) ^ FromKnownSpamAccount(mail)
```

Method: .ive] pseudo-log-likelihood (fast conjunction

□ use prior with mean of 0 and std dev of 5

● Query preds: ○ Evidence preds

Logistic Regression Binary Spam Classification using *pracmln*

pracmln is geared towards practical applicability and ease of integration, and to this end is equipped with a graphical interface and a web application that can be used to easily learn and make inferences from MLNs. As can be seen in the image above, learning MLN weights using *pracmln* is extremely easy and user friendly. The spam classification mentioned in the example can be conceptualised within minutes. Inference from an MLN is equally simple.

However, *pracmln* is written completely in Python, and this project thus aims to make it faster by porting the computationally intensive aspects of the software to Cython.

Goals

Cython is an extension to Python that allows static compilation of Python modules to C libraries. Cython can thus be thought of as a superset of the Python language - the Cython interpreter understands Python, but also understands C-like syntax with static variable typing. This can dramatically improve execution times, and thus help make *pracmln* faster.

This project aims to port the existing computationally intensive portions of *pracmln* - the *learning* and *inference* portions - from Python to Cython. The exact files that will be converted are listed below (listed from the pracmln/python3 folder- corresponding changes will be made in the pracmln/python2 folder too):

```
python3/pracmln/mln/learning
     bpll.py
     cll.py
     common.py
     ll.py
     multidb.py
     optimize.py

pracmln/python3/pracmln/mln/inference
     exact.py
     gibbs.py
     infer.py
     ipfpm.py
     maxwalk.py
     mcmc.py
     mcsat.py
     wcspinfer.py
```

It is worth noting here that no significant work is required to port the `__init__.py` files and the `softeval.py` file to Cython from Python, and hence these have been ignored from this listing. Also, there are two extra files to be converted in the in the `python2/pracmln/mln/learning` folder that aren't in the python3 folder:

```
pracmln/python2/pracmln/mln/learning
    pll.py
    sll.py
```

Thus the deliverables of the project will be the converted **.pyx* Cython files for the entire *learning* and *inference* modules of *pracmln*.

Timeline

I. Pre selection period: March 27 - April 23

- Continue getting familiar with *pracmln*
- Use the Python api to redo the assignment, instead of the GUI *mlnlearn* and *mlnquery* tools. This is just to understand the Python code in *pracmln* further, and test it myself. The current test files seem to just be using the *smokers* and *taxonomies* examples and running various learning and inference tasks upon them. Hopefully, this step will enable me to more confidently write Cython code that is actually equivalent to the Python code it is replacing.
- As practice, convert code from my first Python project (a reduced clone of pastebinit) into Cython - ensuring the code still functions correctly. I have written code directly in Cython in the past, but never had to translate from Python to Cython - and this exercise will help me learn how to do so without causing any change in behaviour.

II. Community bonding period: April 23 - May 13

- Learn the intricacies of the various learning and inference algorithms used in *pracmln*. This would involve learning about a few specific algorithms I am currently not very well versed with such as MC-SAT.
- Review known algorithms from coursework I have completed such as MPE and Gibbs Sampling.
- Learn the functions of the various files in the learning and inference folders with guidance from the organisation mentors. This will help devise a logical sequence in which to convert them. Tentatively (from my exploration so far), I think it would be easier to convert inference algorithms first, and then learning algorithms - for better testing and debugging.

III. Work period 1: May 14 - June 11

During this period I will focus first on the inference algorithms.

- May 14 to May 23: Starting slowly, I will first convert *exact.py* - implementing enumeration ask. I will test this thoroughly and get comfortable with the workflow employed.
- May 24 to June 2: Convert code related to the *MC-SAT* and *Gibbs Sampling* inference algorithms.
- June 3 to June 11: Convert code related to the *MaxWalk-SAT* algorithm, and the *MRF to WCSP conversion* code.

Thus, by the end of this period, for the phase I evaluation, I will try to complete the conversion of the inference modules of *pracmln* to Cython. This may require a little more time to test and debug, depending on the difficulty of the tasks I encounter along the way.

IV. Work period 2: June 12 - July 9

- June 12 to June 21: Test and debug any errors found in the new inference modules of *pracmln*.
- June 22 to July 1: Convert *log likelihood learning* and *pseudo likelihood learning* related code.
- July 2 to July 9: Convert files related to *pseudo-likelihood learning with custom grounding*, and *composite likelihood learning*.

Thus, by the end of this period, for the phase II evaluation, I hope to have completed the conversion of the inference modules, and started on the learning modules.

V. Work period 3: July 10 - August 14

- July 10 to July 19: Convert any converted files (viz. *multidb.py*, and *common.py*).
- July 20 to July 29: Test learning modules written so far.
- July 30 to August 14: Tidy up any backlogged work.

Thus, by the end of the summer, for the final evaluation, I hope to have delivered on all the promised goals mentioned.

VI. Post GSoC timeline: August 15 onwards

I hope to continue contributing towards the development of *pracmln*. Some ideas (which need further discussion) off the top of my head are:

- Exploring the use of continuous integration tools and more rigorous unit testing (TravisCI).
- Writing more detailed documentation, with demo tutorials in the form of Jupyter Notebooks.

Motivation

[This section is the same as in the assignment sent via email]

I have always been interested in applied Machine Learning. I recently came across the Alchemy project online, and began learning about MLNs. This fascinated me, as it married the First Order Logic and Probabilistic Graphical Modelling that I had studied in my AI and ML courses respectively. I was naturally excited when I found out that I could work on pracmln.

While I am not an expert on MLNs, I do have a lot of programming experience that I can contribute to the pracmln project. This way I can practice my Python and Cython skills, and learn about MLNs too. I am extremely interested in this project, and believe that my experience in AI and ML will enable me to get up to speed on the theoretical aspects of MLNs very quickly. I would like to use my knowledge of programming, machine learning, and optimisation to help make pracmln faster.

Thus, I believe I am the right candidate for this project due to my programming aptitude and knowledge about probabilistic models and logic.

Personal Details

Name: Kaivalya Rawal

University: Birla Institute of Technology and Science - Pilani, Goa Campus.

Major: Computer Science (B.E Hons)

Year of Study: Currently in my third year of college. Expect to graduate in 2019.

Relevant Coursework:

- Machine Learning
- Artificial Intelligence
- Data Structures and Algorithms
- Probability and Statistics
- Logic in Computer Science
- Graphs and Networks
- Object Oriented Programming

Other Relevant Experience:

- Teaching assistant for the Machine Learning course in my college.
- Working towards writing an efficient implementation of the Actor-Critic RL algorithm in this paper - in pure Python with static Cython optimisation wherever necessary.
- Contributor to the aima-code project online, thus familiar with Python implementations of AI algorithms.
- Three years of experience using Python, five years using Java.
- Maintain an extensively documented python project on GitHub that intelligently manages Linux terminal history.
- Led a project last year to design and create screencasts to help first-year students prepare for their computer programming labs (in C). Illustrated examples and presented the motivation behind the programming features that they were being taught and evaluated upon in their classes.

Technical Skills:

I usually use Python when writing programs, but am proficient in Java and C too. I have not used Cython professionally, but am familiar with it as I have used it on occasion to speed up tasks in various personal projects. I am comfortable using Bash scripts, the Linux terminal, and Git version control. I have experience in machine learning using Python and R and declarative programming using Prolog.

Links:

- Github: github.com/kaivalyar
- Linkedin: linkedin.com/in/kaivalyar
- Personal Blog: [redacted]

Contact Information:

- Email: [redacted]@gmail.com
- Alternate Email: [redacted]@goa.bits-pilani.ac.in
- Phone: +91 [redacted]
- Alternate phone: +91 [redacted]
- Postal Address: [redacted]

Mentors

Name: Daniel Nyga

Email: [redacted]

Unfortunately, I found out about the pracmln project very late, and thus have not yet been in contact with the mentor. However, I understand GSoC guidelines, and plan to communicate frequently (at least once in three days) through email with updates about the project.

While I didn’t have the chance to have my proposal reviewed before submission, I realise that the requirements of the organisation can change, and am thus very open to modifications in my proposed plan over the summer - to add or modify deliverables. I am extremely motivated towards working for this project, and believe I will be able to rise up to the challenges presented by this project - leading to both improvements in the pracmln software, and learning for me.

This is the **only** project I am applying for in this GSoC as I am very interested in this project, as it directly aligns with my interests and aims in the near future. It allows me the perfect opportunity to use my Python and Cython programming skills to help benefit the open-source statistical relational learning community.