Computer Science Tripos – Part II – Project Proposal

A probabilistic programming language in OCaml

A. Roy, Christ's College

October 17, 2019

Project Originator: Dr. L. Wang

Project Supervisor: Dr R. Mortier

Director of Studies: Dr R. Mortier

Project Overseers: Dr J. A. Crowcroft & Dr T. Sauerwald

Introduction

A probabilistic programming language (PPL) is a framework in which one can create statistical models and have inference run on them automatically. A PPL can take the form of its own language (i.e. a DSL), or be embedded within an existing language (such as OCaml). The ability to write probabilistic programs within OCaml would allow us to leverage the benefits of OCaml, such as expressiveness, a strong type system, and memory safety. The use of a numerical computation library, Owl, will allow us to perform inference performantly.

Starting Point

There do exist PPLs for OCaml, such as IBAL [1], as well as PPLs for other languages, such as WebPPL (JS), Church, Infer.Net to name a few. My PPL can draw on some of the ideas introduced by these languages, particularly in implementing efficient inference engines.

I will be using an existing OCaml numerical computation library (Owl). This library does not contain methods for probabilistic programming in general, although it does contain modules which will help in the implementation of an inference engine such as efficient random number generation and lazy evaluation.

I have experience with the core SML language, which will aid in learning basic OCaml due to similarities in the languages, however I will still have to learn the modules system.

Substance and Structure of the Project

There are 2 main components to the system, namely the modelling API and the inference engine.

Modeling API

The modeling API is used to represent a statistical model. For example, in mathematical notation, a random variable representing a coin flip may be represented as $X \sim N(0, 1)$, but in a PPL we need to represent this as code. An example would be

```
Variable<double> x = Variable.GaussianFromMeanAndVariance(0, 1)
```

in the Infer.Net (F#) language. In OCaml, there will be many different options for representing distributions, and a choice will need to be made about whether to create a separate domain specific language (DSL) or whether to embed the language in OCaml as a library.

Inference Engine

There are many different options for a possible inference engine

Evaluation

The PPL developed here will be compared to existing PPLs for OCaml (in particular, IBAL), comparing performance for programs describing the same models. I will also want to quantify exactly what kind of problems need to be supported by my PPL and make sure these kind of programs can be run. I will also want to support a minimum number of standard distributions, e.g. bernoulli, normal, geometric, etc. or enable users to define custom distributions.

Success Criteria

- **Available distributions**: I will aim to make sure my PPL has at minimum the bernoulli and normal distributions available as basic building blocks to build more complex probabilistic programs.
- **Language Features**: I will aim to support some subset of Ocaml language features, such as 'if' statements, 'for' loops, operators and functions.
- **Performance**: This is a quantitative measure, comparing programs written in my PPL to equivalent programs in other PPLs.
- **Expressiveness**: This is a more qualitative measure, comparing equivalent programs with the same goals across various PPLs.

Extensions

There are several extensions which could be considered, time permitting:

1. There could be more options for the inference engine, i.e. implementing more than one inference algorithm. Different algorithms are suited to different inference tasks, so this would be a worthwhile extension
2. I could add more distributions, as well as the ability to create custom distributions
3. Include the ability to visualise results using the plotting module in owl.
4. Include the ability to visualise the model in which inference is being performed (e.g. the factor graph)

Schedule

Planned starting date is 28/10/19, the Monday after handing in project proposal. Work is broken up into roughly 2 week sections.

Michaelmas Term

- **Weeks 3-4 (28/10/19 – 10/11/19)**
  Set up IDE and local environment - installing Owl and practicing using it. Read papers on past PPLs and implementations, both ocaml or otherwise. Set up project repository and directory structure.
- **Weeks 5-6 (11/10/19 – 24/11/19)**
  Design a basic modelling API and write module/function signatures. Decide how to implement this (e.g. DSL vs library). Research inference algorithms and make sure they will fit into the modelling API designed.
- **Weeks 7-8 (25/10/19 – 08/12/19)**
  Begin to implement the modelling API, and allow running a model 'forward', i.e. generating samples.

Christmas Holidays

- **Weeks 1-2 (09/12/19 – 22/12/19)**
  Begin to implement a basic inference algorithm (such as MCMC) allowing programs to be run 'backwards' to infer parameters.
- **Weeks 3-4 (23/12/19 – 05/01/20)** [*Christmas Break*]
- **Weeks 5-6 (06/01/20 – 19/01/19)**
  Consider extensions if finished early. Begin writing up progress report.

Lent Term

- **Weeks 1-2 (20/01/19 – 02/02/20)**
  Finish progress report as well as any extensions, time permitting.
  Milestone: Progress report deadline (31/01/19)
- **Weeks 3-4 (03/02/20 – 16/02/20)**
  Prepare for the presentation, begin planning the dissertation, particularly the structure and the content I need to write for each section. Begin writing, starting with the first sections (e.g. introduction).
  Milestone: Progress report presentations (06/02/20)
- **Weeks 5-6 (17/02/20 – 01/02/20)**
  Finish writing up the bulk of the implementation section.
- **Weeks 7-8 (02/03/20 – 15/03/20)**
  Complete first draft of dissertation and complete any unfinished tasks.

Easter Holidays

- **Weeks 1-6 (16/03/20 – 26/04/20)**
  Improve dissertation based on supervisor feedback

Easter Term

- **Weeks 1-2 (27/04/20 – 07/04/20)**
  Finalise disseration after proof reading and hand in.
  Milestone: Electronic Submission deadline (08/04/20)

Resources Required

Hardware I intend to use my personal laptop for the main development and subsequent write up (HP Pavilion 15, 8GB RAM, i5-8265U CPU, running Ubuntu and Windows dual booted).

Software The required software includes the ocaml compiler, with a build system (dune) and a package manager (opam). I will also use the IDE VSCode with an OCaml extension, as well as git for version control and latex for the write up.

Backups For backups, I will use GitHub to host my git repository remotely, pushing frequently. I will also backup weekly to a USB stick in case of failures. The software I require is available on MCS machines, so I'll be able to continue work in the event of a hardware failure.

References

[1] Oleg Kiselyov and Chung-Chieh Shan. Embedded probabilistic programming. In *IFIP Working Conference on Domain-Specific Languages*, pages 360–384. Springer, 2009.