Probabilistic models and inference: Project

December 17, 2024

Part I: Introduction

In this first part of the project, your task is to think how to break inference procedures. By now, you have seen several different inference procedures. And you know that there is no single inference procedure that works for every model. Some, as we mentioned during the lectures, tend to work well for more problems than others, but generally you have to put thought into designing the inference procedure for a specific problem. In week 5, you will see how to do that effectively, but that will be the topic for Part 2 of the project.

Part A: problem library

Your first task is to collect a set of probabilistic models. The purpose of this collection is to stress test inference procedures and gain insight into their properties, including strengths and weaknesses.

How do you choose which models to select? The models you chose should allow you to test how effective an inference procedure is on a given model. Therefore, start from the inference procedures we covered in the lectures. What kind of models make it easy for an inference procedure to estimate the right posterior quickly? Which kind of model would make an inference procedure slow in estimating the posterior? Maybe there is a particular structure of a model that makes it nearly impossible to estimate the posterior? Starting from a model you expect an inference procedure to work well on, how do you have to modify it to make it more difficult?

Where do you find the models?

- Invent them! We are not looking for meaningful models, but want you to focus on grasping the properties of the models and how they influence the effectiveness of inference procedures.
- You will find plenty of examples in the documentation of the probabilistic programming language Pyro: https://pyro.ai/examples/
- PPL bench has an interesting collection of problems: https://pplbench.org/
- Bayesian Method for Hackers has a nice bunch of probabilistic programs
- STAN Case Studies publishes case studies with probabilistic programs in STAN

- Modify one program into several ones! We are not asking for models to be unique. You can start from one model and modify it to be more difficult or easy for particular inference procedures.

What to submit? Your submission should be a PDF document containing the following:

- at least two probabilistic models/programs per group member
- for each program, an explanation of why you chose this program/model as a test case. for example, which inference procedures should do well on this models; which inference procedures should do badly; if you modified a model, why did you modify it in this particular way... anything that factored into your reasoning.

  Upload the PDF on Brightspace.

Part B: Translation to Bayesian networks

As you have seen in the first week of the course, probabilistic programs are a generalisation of Bayesian networks. Thinking about probabilistic programs as Bayesian networks is still the best way to understand the complexity of probabilistic models.

Your task consists of two parts:

1. implement a translator from probabilistic programs to Bayesian networks, follow the compilation procedure from van de Meent et al. [2018], Chapter 3.
2. Implement Important Sampling, using the graph-based evaluation from the same chapter.

A few things to note:

1. The translation will give you a graph-structure representation, but not necessarily a Bayesian network.
2. However, the resulting graph will be easy to sample from, regardless of whether the distributions in it are discrete or continuous.
3. We don't expect you to support a full-fledged programming language. You should aim to support only the primitives of a language defined in Chapter 2 of van de Meent et al. [2018], with the exception of:
   - data structures like lists, vectors, and dictionaries
   - definition of new functions within a probabilistic program

4. You can assume a fixed set of built-in functions that mostly do algebraic operations and comparisons over distributions and values coming from those distributions. Let your set of chosen problems dictate what you need.

5. For `for` loop, assume that they are bounded apriori (i.e., you always know how many steps a loop needs to do)

What to submit?

- Upload a zip file with all of your code
- The submission should include a notebook (jupyter, or other) demonstrating how to use your implementation of Importance sampling on a chosen set of problems from Part A

Last few practicalities

- You can use any programming language of your choice. Just make sure we can run your code.
- The tasks are somewhat simprecise *b*y design. You will have to make choices, and we are interested in your thought process behind those decisions.
- The **deadline** is the end of Week 7.

Part II: Introduction

In the second part of the project, you will pick up from where you stopped in Part I.A. You will, however, assume a different role: instead of trying to break the inference procedures, you will now develop custom inference algorithms that perform well on a given problem.

Part A

Start with the problems you have collected for Part I.A. Think of the reasons why you included the models in your benchmark. How can you leverage those to develop a specialised inference procedure that works well on the problem? Don't limit yourself to those reasons only; look into the behaviour of your current method (using the insights from Gelman et al. [2020]) and improve it.

How do we know that one inference procedure is better than another? Well, that is also up to you to define. The standard definition focuses on the rate of convergence – how many samples do we need

to get close to our posterior? You can stick to that, or introduce a more nuanced definition that considers the properties of your problem.

Compare your inference procedure to the out-of-the-box implementations of Importance Sampling, Metropolis-Hastings, and Hamiltonian Monte Carlo from Gen.jl.

Part B

In this part, you will further push the idea of custom inference procedures. During the lectures, we emphasised that no single inference procedure works best for all models. While that is true, the idea of programmable inference gives you the power to construct inference procedures that work well on a large class of problems (if nothing else, then by smartly combining the *basic* inference procedures).

Your task is to develop a custom inference procedures that works, ***on average***, better than the out-of-the-box Importance Sampling, Metropolis-Hasting, and Hamiltonian Monte Carlo on your set of problems. To compute the average performance of an inference method, you should compute your performance metric (e.g., improvement in sample quality) over all of your problems. You goal is to push the curve of your inference procedures above the out-of-the-box ones.

What to submit?

- Upload a zip file with all of your code
- The submission should include a notebook (Jupyter or other) demonstrating how to use your implementations AND a pdf briefly explaining the motivating choices you made when implementing all the inference procedures. Include the chronology – what have you tried, and why were you not satisfied with the performance?

Last few practicalities

- We highly recommend using Gen.jl for this part of the project. You can technically do it in any other probabilistic programming language, but Gen will make the project significantly easier for you.
- The tasks are somewhat imprecise *b*y design. You will have to make choices, and we are interested in your thought process behind those decisions.
- The **deadline** is the end of Week 9.

References

Andrew Gelman, Aki Vehtari, Daniel Simpson, Charles C. Margossian, Bob Carpenter, Yuling Yao, Lauren Kennedy, Jonah Gabry, Paul-Christian BÃijrkner, and Martin ModrÃąk. Bayesian workflow. *arXiv stat.ME*, 2011(01808), November 2020. DOI: 10.48550/arXiv.2011.01808. URL https://arxiv.org/abs/2011.01808.

Jan-Willem van de Meent, Brooks Paige, Hongseok Yang, and Frank Wood. An introduction to probabilistic programming, 2018. URL http://arxiv.org/abs/1809.10756. cite arxiv:1809.10756Comment: Under review at Foundations and Trends in Machine Learning.