Extend Variational Inference Methods in PyMC3

Abstract

Variational inference is a great approach for doing really complex, often intractable bayesian inference in approximate form. Common methods (e.g. ADVI) lack from complexity so that approximate posterior does not reveal the true nature of underlying problem. In some applications it can yield unreliable decisions.

Recently on NIPS 2017 OPVI framework was presented. It generalizes variational inverence so that the problem is build with blocks. The first and essential block is Model itself. Second is Approximation, in some cases $logQ(D)$ is not really needed. Necessity depends on the third and forth part of that black box, Operator and Test Function respectively.

Operator is like an approach we use, it constructs loss from given Model, Approximation and Test Function. The last one is not needed if we minimize KL Divergence from Q to posterior. As a drawback we need to compute $loqQ(D)$. Sometimes approximation family is intractable and $loqQ(D)$ is not available, here comes LS(Langevin Stein) Operator with a set of test functions.

Test Function has more unintuitive meaning. It is usually used with LS operator and represents all we want from our approximate distribution. For any given vector based function of z LS operator yields zero mean function under posterior. $loqQ(D)$ is no more needed. That opens a door to rich approximation families as neural networks.

Not only ADVI and Langevin Stein Operator VI are applicable with OPVI framework. Normalizing, Householder Flows fit well for it.

Motivation

My recent contributions (Implementing OPVI) to PyMC3 created a good basis for extending variational inference in PyMC3 even further. I tried to transfer theoretical framework to python code and it succeed. Now main logic is in base classes and all rotines are abstracted and use public interface that is provided by developer. Implementing state-of-the-art methods is now not a challenge, you should just break the problem into 4 blocks described above and implement abstract methods.

I also have a side project Gelato for using pymc3 in neural networks. So my future plans are the following:

1. Implement Normalizing Flows
2. Implement Householders Flows
3. Implement Langein Stein Operator

4. Integrate OPVI to Gelato

Technical Details

I'm going to use the following libraries:

- **Theano**
- **PyMC3**
- **Gelato**
- **Lasagne**
- **NumPy**

As support material I'll use papers from arXiv:

Variational Inference with Normalizing Flows Improving Variational Auto-Encoders using Householder Flow Operator Variational Inference

Schedule of Deliverables

May 1th - May 28th, Community Bonding Period

Integrate OPVI to Gelato

Work on documentation for OPVI and Histogram. Add a noteboook with comprehensive example using Gelato.

May 29th - June 10th

Implement Normalizing and Householder Flows

These flows have similar structure so implementing them both at once will be the best decision.

June 11th - June 23th

Debug Flows, refine code, improve performance.

July 3rd - July 7th

Add documentation for Flows

July 10th - July 23d End of Phase 2

Implement Langevin Stein Operator with convergence tests

Not all tests are supposed to be passed by the end of the Phase 2, making it work is a tough problem according to my past experience

July 24th - July 28th, Begin of Phase 3

Further work on making LS Op work

July 31st - August 4th

Finish LS Op

August 7th - August 10th

Add documentaton for Langevin Stein Operator

August 11th - August 18th

Write a notebook with example of LS Op

August 21st - August 25th, Final Week

Byesian Summer School, not sure I can work here

August 28th - August 29th, Submit final work

Submit

Future works

Read arXiv, collect ideas

Development Experience

Yandex Analyst-Developer Intern (summer 2016), PyMC3 developer

Other Experiences

Yandex Data Factory Analyst Intern (now)

Why this project?

I'm a great fan of bayesian statistics and see it is usefull for many practical applications. I also love development and good codestyle. This project is interesting for me from both point of views. I'm also planning to use my results for my research projects and work.

Links

GitHub GSoC PR, Issue