Pyro Documentation

Release 0.1.2-7-gabe15cc

Uber AI Labs

Dec 07, 2018

Contents:

CHAPTER 1

Installation

1.1 Install from Source

Pyro supports Python 2.7.* and Python 3. To setup, install PyTorch then run:

```
pip install pyro-ppl
```

or install from source:

```
git clone https://github.com/uber/pyro.git
cd pyro
python setup.py install
```

Warning: Some bleeding-edge features of Pyro (e.g. *enum_discrete*) require a very recent version of PyTorch. We recommend installing PyTorch from source using the *master* branch. See PyTorch install instructions for details.

CHAPTER 2

Getting Started

- Install Pyro.
- Learn the basic concepts of Pyro: models and inference.
- Dive in to other tutorials and examples.

CHAPTER 3

Primitives

CHAPTER 4

Inference

In the context of probabilistic modeling, learning is usually called inference. In the particular case of Bayesian inference, this often involves computing (approximate) posterior distributions. In the case of parameterized models, this usually involves some sort of optimization. Pyro supports multiple inference algorithms, with support for stochastic variational inference (SVI) being the most extensive. Look here for more inference algorithms in future versions of Pyro.

See Intro II for a discussion of inference in Pyro.

4.1 SVI

4.2 ELBO

4.3 Importance

4.4 Search

CHAPTER 5

Distributions

5.1 Primitive Distributions

5.1.1 Bernoulli

5.1.2 Beta

5.1.3 Categorical

5.1.4 Cauchy

5.1.5 Delta

5.1.6 Normal

5.1.7 Exponential

5.1.8 Gamma

5.1.9 HalfCauchy

5.1.10 LogNormal

5.1.11 Multinomial

5.1.12 Poisson

5.1.13 Uniform

5.2 Transformed Distribution

5.2.1 TransformedDistribution

5.2.2 Bijector

CHAPTER 6

Parameters

Parameters in Pyro are basically thin wrappers around PyTorch Variables that carry unique names. As such Parameters are the primary stateful objects in Pyro. Users typically interact with parameters via the Pyro primitive *pyro.param*. Parameters play a central role in stochastic variational inference, where they are used to represent point estimates for the parameters in parameterized families of models and guides.

6.1 ParamStore

CHAPTER 7

Neural Network

The module *pyro.nn* provides implementations of neural network modules that are useful in the context of deep probabilistic programming. None of these modules is really part of the core language.

7.1 AutoRegressiveNN

CHAPTER 8

Optimization

The module *pyro.optim* provides support for optimization in Pyro. In particular it provides *PyroOptim*, which is used to wrap PyTorch optimizers and manage optimizers for dynamically generated parameters (see the tutorial SVI Part I for a discussion). Any custom optimization algorithms are also to be found here.

8.1 PyroOptim

8.2 ClippedAdam

CHAPTER 9

Advanced Features

Beneath the built-in inference algorithms, Pyro has a library of flexible primitives for creating new inference algorithms and working with probabilistic programs. The core abstraction of composable inference in Pyro is the *poutine* (short for Pyro Coroutine). Pyro's inference algorithms are all built by applying *poutine* s to stochastic functions.

9.1 Poutines (Pyro Coroutines)

9.1.1 Poutine

9.1.2 BlockPoutine

9.1.3 ConditionPoutine

9.1.4 EscapePoutine

9.1.5 IndepPoutine

9.1.6 LiftPoutine

9.1.7 ReplayPoutine

9.1.8 ScalePoutine

9.1.9 Trace

9.1.10 TracePoutine

CHAPTER 10

Indices and tables

- genindex
- search