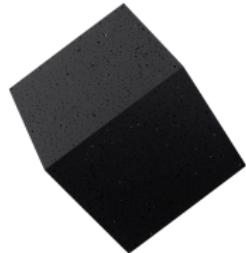


Why Aren't You Using Probabilistic Programming?

Dustin Tran
Columbia University





Alp Kucukelbir



Adjji Dieng



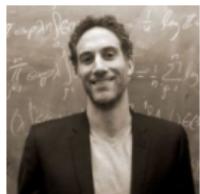
Dave Moore



Dawen Liang



Eugene Brevdo



Ian Langmore



Josh Dillon



Maja Rudolph



Brian Patton



Srinivas Vasudevan



Alex Alemi



Matt Hoffman



David Blei



Kevin Murphy



Rif Saurous

What is probabilistic programming?

Probabilistic programs reify models from mathematics to physical objects.

- Each model is equipped with memory (“bits”, floating point, storage) and computation (“flops”, scalability, communication).

Anything you do lives in the world of probabilistic programming.

- Any computable model.
- Any computable inference algorithm.
- Any computable application.

Simulation hypothesis.

“The universe is a simulation from a computer program.”

(Zuse, Schmidhuber, Bostrom, Musk)

Inference in a probabilistic program

```
(trace, weight) = query(program, args, observations)
```





home source downloads docs packages blog community ecosystems le juliacon

On Machine Learning and Programming Languages

06 Dec 2017

[openai / pixel-cnn](#)

Code Issues Pull requests Projects Wiki Insights

python3 / Tensorflow implementation of PixelCNN++, as described in "PixelCNN++: A PixelCNN I

discretized Logistic Mixture Likelihood and Other Modifications" <https://arxiv.org/abs/1701.0551>

58 commits 3 branches 0 releases 5 contributors

Branch: master New pull request Create new file Upload files Find file

[openai / iaf](#)

Code Issues Pull requests Projects Wiki Insights

Code for reproducing key results in the paper "Improving Variational Inference with Inverse Autore-

[ps://arxiv.org/abs/1606.04934](https://arxiv.org/abs/1606.04934)

18 commits 2 branches 0 releases 2 contributors

Branch: master New pull request Create new file Upload files Find file

[dpkingma / Merge pull request #12 from openai/revert-8-tf_utils_bugfix](#)

graphy

[carpedm20 / DCGAN-tensorflow](#)

Watch 191

Unstar 3,291

Fork

Code Issues Pull requests Projects Wiki Insights

A tensorflow implementation of "Deep Convolutional Generative Adversarial Networks" <http://carpedm20.github.io/faces/>

tensorflow dcgan gan generative-model

263 commits

2 branches

0 releases

32 contributors

MIT

Branch: master New pull request

Create new file Upload files Find file Clone or download

[carpedm20 / Merge pull request #208 from johnhany/Fix-filename-time](#)

Latest commit 251aa44 25 d

[assets](#) Rename test_***.png files

4 mon

[junyanz / CycleGAN](#)

Watch 264 Unstar 5,000

Code Issues Pull requests Projects Wiki Insights

Software that can generate photos from paintings, turn horses into zebras, perform style transfer, and more.

computer-vision computer-graphics gan dcgan generative-adversarial-network deep-learning image-generation image-improvement

cyclegan pix2pix

80 commits 1 branch 0 releases 8 contributors

Branch: master New pull request

Create new file Upload files Find file

[junyanz / Update README.md](#)

Latest commit

[hanhanggit / StackGAN](#)

Watch 52 Star 935

Code Issues Pull requests Projects Wiki Insights

No description, website, or topics provided.

13 commits 1 branch 0 releases 1 contributor

Branch: master New pull request

Create new file Upload files Find file

[hanhanggit / Update README.md](#)

Latest commit 3

Data

first commit

demo

first commit

The Myth of Probabilistic Programming

**Programming is infeasible if a core operation
in the language is NP-hard.**

For high-dimensional problems + modern probabilistic models, we haven't solved automated inference.

[Code](#)[Issues 117](#)[Pull requests 23](#)[Insights](#)

A library for probabilistic modeling, inference, and criticism. Deep generative models, variational inference. Runs on TensorFlow. <http://edwardlib.org>

[bayesian-methods](#) [deep-learning](#) [machine-learning](#) [data-science](#) [tensorflow](#) [neural-networks](#) [statistics](#) [probabilistic-programming](#)

1,761 commits

19 branches

27 releases

66 contributors

Branch: [master](#) ▾[New pull request](#)[Find file](#)[Clone or download](#) ▾

 christopherlovell committed with dustinvtran fixed invgamma_normal_mh example (#793) ...	Latest commit 081ea53 23 days ago
 docker Use Observations and remove explicit storage of data files (#751)	3 months ago
 docs Revise docs to enable spaces in filepaths; update travis with tf==1.4...	26 days ago

[Sign Up](#)[Log In](#)[all categories](#) ▾[Latest](#)[Top](#)

Topic

Category

Users

Replies

Views

Activity

Iterative estimators ("bayes filters") in Edward?



5

21

7h

Tutorial for multiple variational methods using Poisson regression?



2

20

1d

blei-lab/edward A library for probabilistic modeling, inference, and criticism. <http://edwardlib.org>

Faez Shakil @faezs Jan 23 02:47
Hi @dustinvtran, thanks for edward, the library and surrounding literature have been immense fun to get into. Would you be able to tell me whether it'd be relatively painless to get the inference compute graphs from Ed as native tensorflow graphdefs and use them on mobile platforms? Or would I have to port a bunch of custom ops

PEOPLE REPO INFO



We have an active community of several thousand users & many contributors.

Language: Computational Graphs w/ Random Variables

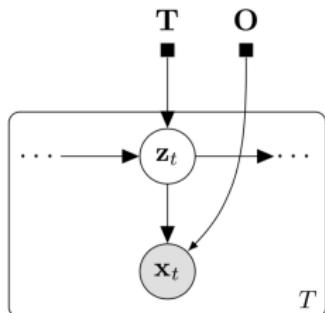
Edward's language augments computational graphs with an abstraction for random variables. Each random variable \mathbf{x} is associated to a tensor \mathbf{x}^* , $\mathbf{x}^* \sim p(\mathbf{x} | \theta^*)$.

```
1 # univariate normal
2 Normal(loc=0.0, scale=1.0)
3 # vector of 5 univariate normals
4 Normal(loc=tf.zeros(5), scale=tf.ones(5))
5 # 2 x 3 matrix of Exponentials
6 Exponential(rate=tf.ones([2, 3]))
```

Unlike `tf.Tensors`, `ed.RandomVariables` carry an explicit density with methods such as `log_prob()` and `sample()`.

For implementation, we wrap all TensorFlow Distributions and call `sample` to produce the associated tensor.

Language Example

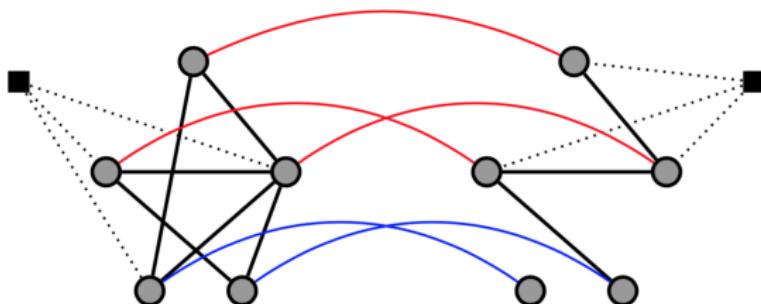


```
1 from edward.models import Normal  
2  
3 T = tf.Variable(tf.random_normal([K, K]))  
4 O = tf.Variable(tf.random_normal([K, D]))  
5 z = x = []  
6 z[0] = Normal(loc=tf.zeros(K), scale=tf.ones(K))  
7 x[0] = Normal(loc=tf.matmul(z[0], 0), scale=0.5)  
8 for t in range(1, T):  
9     z[t] = Normal(loc=tf.matmul(z[t - 1], T), scale=0.1)  
10    x[t] = Normal(loc=tf.matmul(z[t], 0), scale=0.5)
```

State space model for sequences $\mathbf{x} = [\mathbf{x}_1, \dots, \mathbf{x}_T] \in \mathbb{R}^{T \times D}$.

Edward's language enables a *calculus* on random variables.

Inference as Stochastic Graph Optimization



All Inference has (at least) two inputs:

1. red aligns latent variables and posterior approximations;
2. blue aligns observed variables and realizations.

```
inference = ed.Inference({beta: qbeta, z: qz}, data={x: x_train})
```

Inference has class methods to finely control the algorithm. Edward is as fast as handwritten TensorFlow at runtime.

Composable & Hybrid Inference

```
1 n_samples = n_iter_per_epoch * n_epoch
2 qz = Empirical(tf.Variable(tf.random_normal([n_samples, mnist.train.size, K])))
3
4 inference_e = ed.SGHMC({z: qz}, data={x: x_train})
5 inference_e.initialize()
6
7 inference_m = ed.MAP(data={x: x_train, z: qz.params[inference_e.t]})
```

```
8 optimizer = tf.train.RMSPropOptimizer(1e-2, epsilon=1.0)
9 inference_m.initialize(optimizer=optimizer)
```

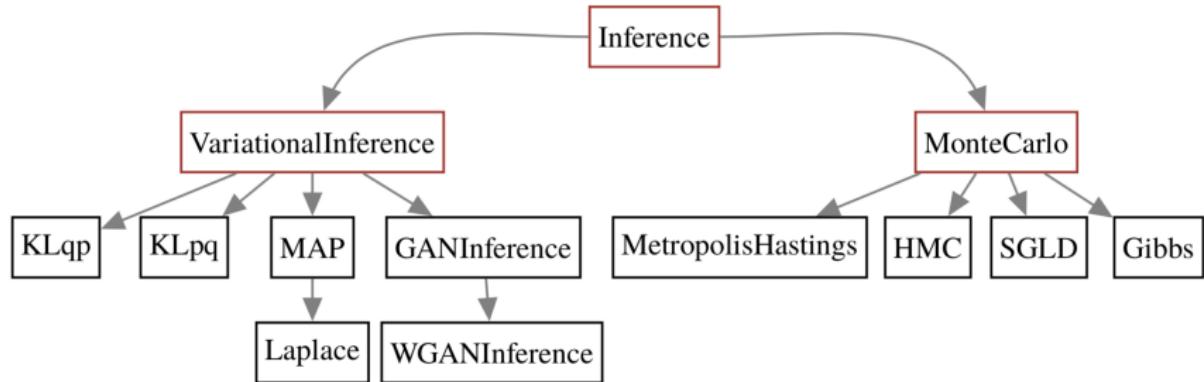
```
inference_z1 = ed.KLpq({beta: qbeta, z1: qz1}, {x1: x1_train})
inference_z2 = ed.KLpq({beta: qbeta, z2: qz2}, {x2: x2_train})
...
for _ in range(10000):
    inference_z1.update()
    inference_z2.update()
```

[Neal & Hinton 1993; Minka 2001; Gelman+ 2017; Hasenclever+ 2015]

Non-Bayesian Inference

```
1 def generative_network(eps):
2     h = Dense(256, activation='relu') (eps)
3     return Dense(28 * 28, activation=None) (h)
4
5 def discriminative_network(x):
6     h = Dense(28 * 28, activation='relu') (x)
7     return Dense(h, activation=None) (1)
8
9 # Probabilistic model
10 eps = Normal(loc=tf.zeros([N, d]), scale=tf.ones([N, d]))
11 x = generative_network(eps)
12
13 inference = ed.GANInference(data={x: x_train},
14                               discriminator=discriminative_network)
15 inference.run()
```

Taxonomy of Inference



Nodes are Edward classes. Arrows denote inheritance.

Why You Shouldn't Use Probabilistic Programming

- **Object-oriented inference** has a high cognitive burden.
- Sometimes it's easier to **build the loss function** than it is to build the program. (e.g. autoregressive models)
- **Programmable inference is hard.** Matt and I spent a year on covering use cases. But we didn't cover all of them.

Why You Will Use Probabilistic Programming

Distributed, Compiled, Accelerated Systems



Probabilistic programming over multiple machines. XLA compiler optimization and TPUs. More flexible programmable inference.

Dynamic Graphs



Probabilistic Torch is a library for deep generative models that extends [PyTorch](#). It is similar in spirit and design goals to [Edward](#) and [Pyro](#), sharing many design characteristics with the latter.

The design of Probabilistic Torch is intended to be as PyTorch-like as possible. Probabilistic Torch models are written just like you would write any PyTorch model, but make use of three additional constructs:

Distributions Backend

```
def pixelcnn_dist(params, x_shape=(32, 32, 3)):  
    def _logit_func(features):  
        # single autoregressive step on observed features  
        logits = pixelcnn(features)  
        return logits  
    logit_template = tf.make_template("pixelcnn", _logit_func)  
    make_dist = lambda x: tfd.Independent(tfd.Bernoulli(logit_template(x)))  
    return tfd.Autoregressive(make_dist, tf.reduce_prod(x_shape))  
  
x = pixelcnn_dist()  
loss = -tf.reduce_sum(x.log_prob(images))  
train = tf.train.AdamOptimizer().minimize(loss) # run for training  
generate = x.sample() # run for generation
```

TensorFlow Distributions consists of a large collection of distributions.
Bijector enable efficient, composable manipulation of probability distributions.

Pytorch PPLs are consolidating on a backend for distributions.

References



edwardlib.org

- Edward: A library for probabilistic modeling, inference, and criticism.
arXiv preprint arXiv:1610.09787, 2016.
- Deep probabilistic programming.
International Conference on Learning Representations, 2017.
- TensorFlow Distributions.
arXiv preprint arXiv:1711.10604, 2017.