Edward: a library for probabilistic modeling, inference, and criticism.

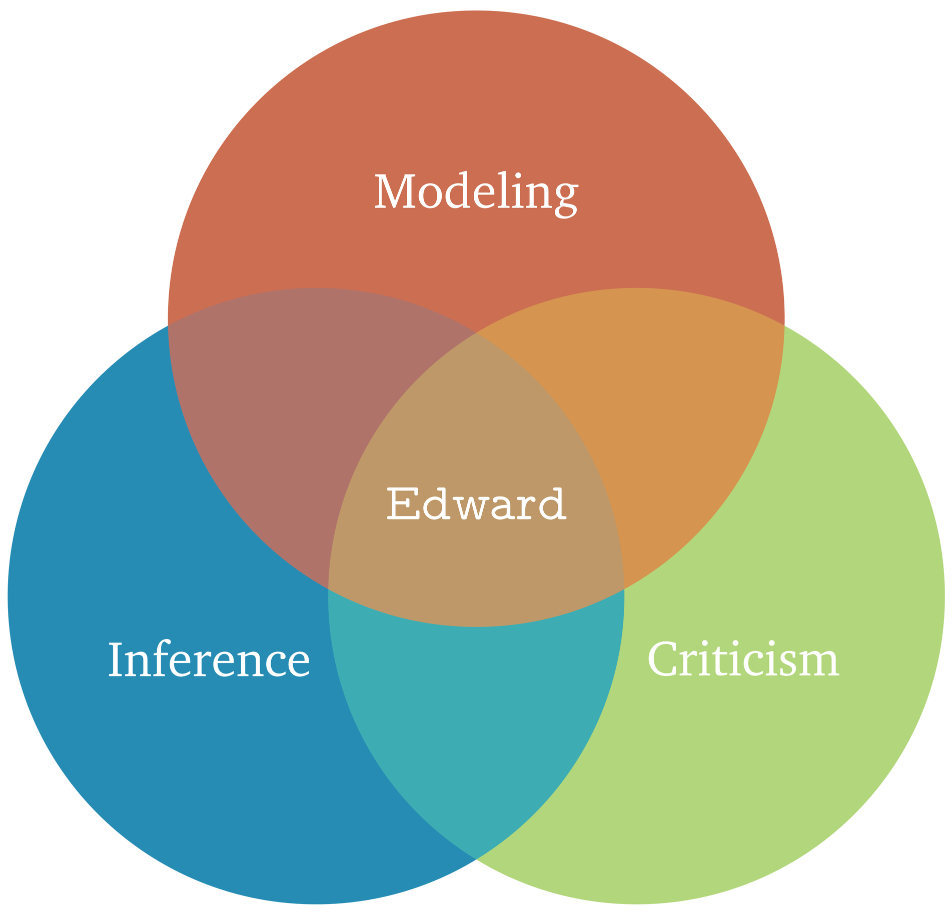
Dustin Tran, David Blei, Alp Kucukelbir, Adji Dieng, Maja Rudolph, and Dawen Liang
Columbia University

Summary

- Edward is a library for probabilistic modeling, inference, and criticism [1].
- Edward supports probability models $p(\mathbf{x}, \mathbf{z})$.
- Edward leverages black box variational inference.
- Edward enables model and inference criticism.
- Edward is a Python/TensorFlow project.

https://github.com/blei-lab/edward

Goals



- Edward is an open-source research library for probabilistic programming research.
- Edward follows Box's philosophy of statistics and machine learning [2]

1. Build a probabilistic model of the process
2. Reason about the process given model and data
3. Criticize the model, revise and repeat

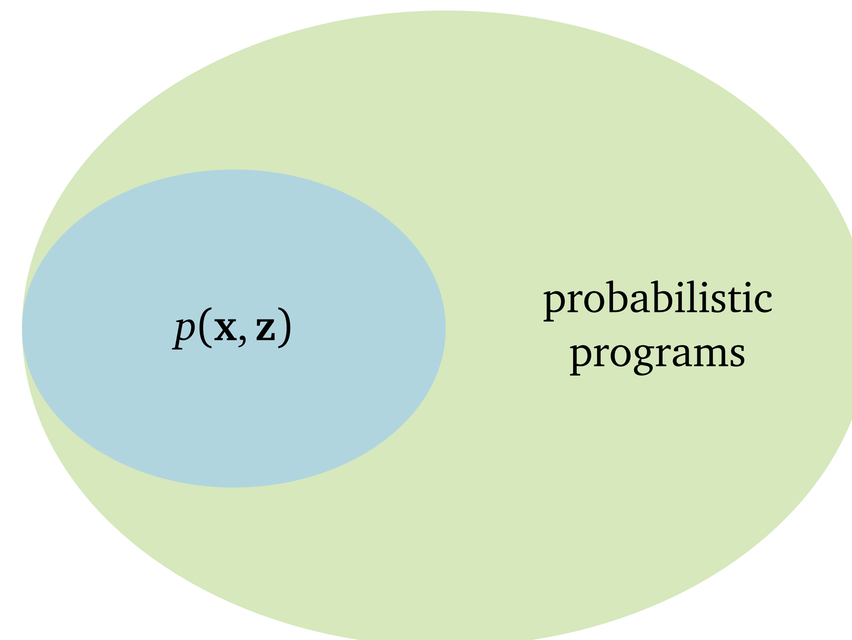
Features

- Edward supports the following modeling languages
  - TensorFlow (with neural network composition via Keras, Pretty Tensor, or TensorFlow-Slim)
  - Stan
  - PyMC3
  - Python through Numpy/Scipy
- Edward implements black box inference through variational inference
  - black box variational inference [3]
  - data-level stochastic variational inference [4]
  - variational auto-encoders [5]
  - delta function / MAP approximation
  - Laplace approximation
- Edward supports model and inference criticism
  - posterior predictive checks [6]
  - a library of evaluation metrics

Backend

- Edward is built on top of TensorFlow
  - computation graphs
  - parallelization / GPU support
  - automatic differentiation
  - optimization algorithms
- Edward implements its own math/probability library.

Scope



- Edward focuses on probability models $p(\mathbf{x}, \mathbf{z})$.
- Edward supports models with
  - large data $\mathbf{x}$
  - continuous or discrete latent variables $\mathbf{z}$
  - complex structures: e.g. hierarchical models, neural networks, deep exponential families.
- The goal is to infer the posterior $p(\mathbf{z} \mid \mathbf{x})$.

Design

Data

Edward `Data` objects are containers. A `Data` object has structure (dimensions); these must match a probability model during inference. A `Data` object may optionally implement a custom subsampling routing; the default is to subsample along the first dimension.

Models

Edward has two types of `Models`:

1. Probability models of data and latent variables
2. Variational models of latent variables

Probability models must implement

`log_prob(self, x, theta)`

Variational models must implement

`sample(self, size=1)` and `entropy(self)`

Inference

Edward supports many forms of variational inference. One form matches the variational model to the posterior $p(\mathbf{z} \mid \mathbf{x})$ by maximizing

$$\text{ELBO} = \mathbb{E}_{q(\mathbf{z}\,;\,\boldsymbol{\lambda})}[\log p(\mathbf{x}, \mathbf{z}) - \log q(\mathbf{z}\,;\,\boldsymbol{\lambda})]. \quad (1)$$

Edward solves this optimization using automatic differentiation and stochastic gradient methods.

Criticism

Edward provides building blocks for criticizing both model and inference. An example is a pipeline for simulating new datasets using samples from the variational approximation; this enables posterior predictive checks.

Smart Inference

Solving Equation (1) is the main computational task of inference. While automatic differentiation helps by avoiding manual derivations of gradients, the expectation in the objective function poses a greater challenge.

There are two approaches to computing gradients of eq. (1)

1. score function estimator (more general)
2. reparameterization estimator (less noisy)

Edward prefers reparameterization, if the variational model admits it. Otherwise, it defaults to the score function estimator.

Edward also prefers an analytic (closed-form) entropy term $\mathbb{E}_{q(\mathbf{z}\,;\,\boldsymbol{\lambda})}[-\log q(\mathbf{z}\,;\,\boldsymbol{\lambda})]$, if the variational model admits it.

Example *(abridged)*

```python
import edward as ed
import tensorflow as tf
import numpy as np

from edward.models import Variational, Dirichlet, Normal, InvGamma
from edward.stats import dirichlet, invgamma, multivariate_normal, norm
from edward.util import get_dims

class MixtureGaussian:

    def __init__(self, K, D):
        self.K = K
        self.D = D
        self.num_vars = (2*D + 1) * K

        self.a = 1
        self.b = 1
        self.c = 10
        self.alpha = tf.ones([K])

    def log_prob(self, xs, zs):
        """Returns a vector [log p(xs, zs[1,:]), ..., log p(xs, zs[S,:])]."""
        N = get_dims(xs)[0]
        pi, mus, sigmas = zs
        log_prior = dirichlet.logpdf(pi, self.alpha)
        log_prior += tf.reduce_sum(norm.logpdf(mus, 0, np.sqrt(self.c)), 1)
        log_prior += tf.reduce_sum(invgamma.logpdf(sigmas, self.a, self.b), 1)

        # Loop over each mini-batch zs[b,:]
        log_lik = []
        n_minibatch = get_dims(zs[0])[0]
        for s in range(n_minibatch):
            log_lik_z = N*tf.reduce_sum(tf.log(pi), 1)
            for k in range(self.K):
                log_lik_z += tf.reduce_sum(multivariate_normal.logpdf(xs,
                    mus[s, (k*self.D):((k+1)*self.D)],
                    sigmas[s, (k*self.D):((k+1)*self.D)]))

            log_lik += [log_lik_z]

        return log_prior + tf.pack(log_lik)

ed.set_seed(42)
x = np.loadtxt('data/mixture_data.txt', dtype='float32', delimiter=',')
data = ed.Data(tf.constant(x, dtype=tf.float32))

model = MixtureGaussian(K=2, D=2)
variational = Variational()
variational.add(Dirichlet(model.K))
variational.add(Normal(model.K*model.D))
variational.add(InvGamma(model.K*model.D))

inference = ed.MFVI(model, variational, data)
inference.run(n_iter=500, n_minibatch=5, n_data=5)
```

Next Steps

Data

- distributed / data in the cloud
- streaming data

Models

- a new modeling language

Inference

- subsampling of latent variables
- amortized variational inference
- marginal maximum likelihood
- alternative divergence measures

Criticism

- library of built-in predictive checks

License

References

[1] Dustin Tran, David M. Blei, Alp Kucukelbir, Adji Dieng, Maja Rudolph, and Dawen Liang. Edward: A library for probabilistic modeling, inference, and criticism, 2016.

[2] George EP Box. Science and statistics. *Journal of the American Statistical Association*, 71(356):791–799, 1976.

[3] Rajesh Ranganath, Sean Gerrish, and David Blei. Black box variational inference. In *Artificial Intelligence and Statistics*, 2014.

[4] Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley. Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347, 2013.

[5] Diederik Kingma and Max Welling. Auto-encoding variational bayes. *arXiv:1312.6114*, 2013.

[6] Andrew Gelman, Xiao-Li Meng, and Hal Stern. Posterior predictive assessment of model fitness via realized discrepancies. *Statistica sinica*, 6(4):733–760, 1996.