

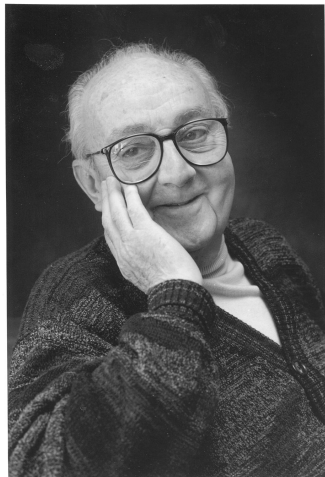
# Probabilistic programming with Edward

John Reid

Biostatistics Unit,  
School of Clinical Medicine,  
Cambridge University

April 25, 2017

# George E.P. Box (1919 - 2013)

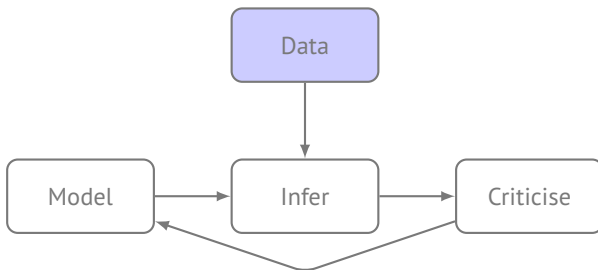


An iterative process for science:

1. Build a model of the science
2. Infer the model given data
3. Criticize the model given data

Box and Hunter, 1962, 1965; Box and Hill, 1967; Box, 1976, 1980

# Box's Loop



Edward is a library designed around this loop.  
(Box, 1976, 1980; David M. Blei, 2014)

# Model comparisons

Inference method	Negative log-likelihood
VAE ( <a href="#">Kingma &amp; Welling, 2014</a> )	$\leq 88.2$
VAE without analytic KL	$\leq 89.4$
VAE with analytic entropy	$\leq 88.1$
VAE with score function gradient	$\leq 87.9$
Normalizing flows ( <a href="#">Rezende &amp; Mohamed, 2015</a> )	$\leq 85.8$
Hierarchical variational model ( <a href="#">Ranganath et al., 2016b</a> )	$\leq 85.4$
Importance-weighted auto-encoders ( $K = 50$ ) ( <a href="#">Burda et al., 2016</a> )	$\leq 86.3$
HVM with IWAE objective ( $K = 5$ )	$\leq 85.2$
Rényi divergence ( $\alpha = -1$ ) ( <a href="#">Li &amp; Turner, 2016</a> )	$\leq 140.5$

**Table 1:** Inference methods for a probabilistic decoder on binarized MNIST. The Edward PPL makes it easy to experiment with many algorithms.

**Edward** is a probabilistic programming language, designed for fast experimentation and research (Tran et al., 2017).

### *Modelling*

- ▶ Composable Turing-complete language of random variables.
- ▶ Examples: Graphical models, neural networks, probabilistic programs.
- ▶ Many data types, tensor vectorization, broadcasting, 3rd party support.

### *Inference*

- ▶ Composable language for hybrids, message passing, data subsampling.
- ▶ Examples: Black box VI, Hamiltonian MC, stochastic gradient MCMC, generative adversarial networks.
- ▶ Infrastructure to develop your own algorithms.

### *Criticism*

- ▶ Examples: Scoring rules, hypothesis tests, predictive checks.

Built on TensorFlow (features distributed computing, GPUs, autodiff).

```
# DATA
```

```
x_data = np.array([0, 1, 0, 0, 0, 0, 0, 0, 0, 1])
```

```
# MODEL
```

```
p = Beta(a=1.0, b=1.0)
```

```
x = Bernoulli(p=tf.ones(10) * p)
```

```
# VARIATIONAL DISTRIBUTION
```

```
qp_a = tf.nn.softplus(tf.Variable(tf.random_normal([])))
```

```
qp_b = tf.nn.softplus(tf.Variable(tf.random_normal([])))
```

```
qp = Beta(a=qp_a, b=qp_b)
```

```
# INFERENCE
```

```
inference = ed.KLqp({p: qp}, data={x: x_data})
```

```
inference.run(n_iter=500)
```

```
# CRITICISM
```

```
x_post = ed.copy(x, {p : qp})
```

```
def T(xs, zs):
```

```
    return tf.reduce_mean(xs[x_post])
```

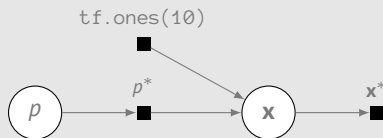
```
ed.ppc(T, data={x_post: x_data})
```

## Model code

```
p = Beta(a=1.0, b=1.0)
x = Bernoulli(p=tf.ones(10) * p)
```

The random variables  $p$  and  $x$  are represented by tensors  $p^*$  and  $x^*$  in the tensorflow computational graph

## Computational graph



Random variables are equipped with methods for likelihoods  $\log(x|p)$ , expectations  $\mathbb{E}_{p(x|p)}[x]$ , and sampling  $\sim p(x|p)$ .

Graph can be executed by `x.value()` which returns the tensor  $x^*$  and simulates the generative process.

# Model construction

Key concept is compositionality:

- ▶ Computational graphs can contain arbitrary tensorflow constructs
- ▶ Tensorflow conditional evaluations permit nonparametric processes
- ▶ Interface with third party tensorflow libraries, e.g. Keras for deep learning

## Deep generative model

```
from edward.models import Bernoulli, Normal
from keras.layers import Dense

z = Normal(mu=tf.zeros([N, d]), sigma=tf.ones([N, d]))
h = Dense(256, activation='relu')(z.value())
x = Bernoulli(logits=Dense(28 * 28)(h))
```



# Inference abstraction

Edward's random variables can represent probabilistic models as computational graphs.

How to perform inference? We desire:

- ▶ Support for many inference classes
- ▶ The posterior can be further composed as part of a larger model

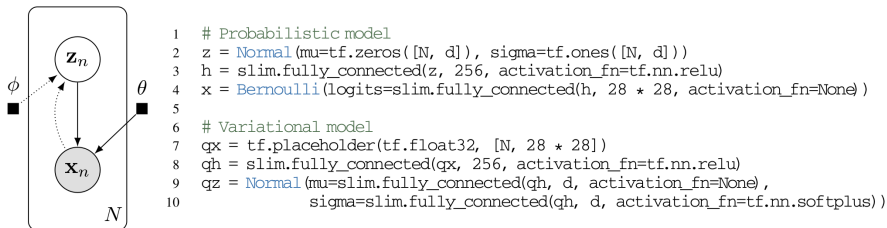
Edward abstracts this as an optimisation problem

$$\min_{\lambda, \theta} \mathcal{L}(p(\mathbf{z}, \beta \mid \mathbf{x}_{\text{train}}; \theta), q(\mathbf{z}, \beta; \lambda)),$$

where  $q$  can be a variational distribution, point estimate or collection of samples.

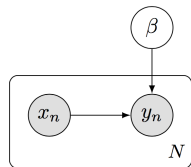
The loss for the optimisation problem is represented in the same computational graph as the model.

# MNIST variational auto-encoder



**Figure 2:** Variational auto-encoder for a data set of  $28 \times 28$  pixel images: (left) graphical model, with dotted lines for the inference model; (right) probabilistic program, with 2-layer neural networks.

# GPU-accelerated Hamiltonian Monte Carlo



```
1  # Model
2  x = tf.Variable(x_data, trainable=False)
3  beta = Normal(mu=tf.zeros(D), sigma=tf.ones(D))
4  y = Bernoulli(logits=tf.dot(x, beta))
5
6  # Inference
7  qbeta = Empirical(params=tf.Variable(tf.zeros([T, D])))
8  inference = ed.HMC({beta: qbeta}, data={y: y_data})
9  inference.run(step_size=0.5 / N, n_steps=100)
```

Bayesian logistic regression for the Covertypes dataset ( $N = 581012$ ,  $D = 54$ ). 12-core Intel i7-5930K CPU at 3.50GHz and a NVIDIA Titan X (Maxwell) GPU. 100 iterations of HMC with same settings.

Probabilistic programming language	Runtime
Stan (1 CPU)	171 sec
PyMC3 (12 CPU)	361 sec
<b>Edward (12 CPU)</b>	<b>8.2 sec</b>
<b>Edward (GPU)</b>	<b>4.9 sec</b> (35x faster than Stan)

(Carpenter et al., 2017; Salvatier, Wiecki, and Fonnesbeck, 2015)

# References I

- [1] G. E. P. Box and William G. Hunter. “A Useful Method for Model-Building”. In: *Technometrics* 4.3 (1962), pp. 301–318. JSTOR: 1266570.
- [2] G. E. P. Box and William G. Hunter. “The Experimental Study of Physical Mechanisms”. In: *Technometrics* 7.1 (1965), pp. 23–42. JSTOR: 1266125.
- [3] G. E. P. Box and W. J. Hill. “Discrimination among Mechanistic Models”. In: *Technometrics* 9.1 (1967), pp. 57–71. JSTOR: 1266318.
- [4] George E. P. Box. “Science and Statistics”. In: *Journal of the American Statistical Association* 71.356 (1976), pp. 791–799. JSTOR: 2286841.
- [5] George E. P. Box. “Sampling and Bayes’ Inference in Scientific Modelling and Robustness”. In: *Journal of the Royal Statistical Society. Series A (General)* 143.4 (1980), pp. 383–430. JSTOR: 2982063.
- [6] David M. Blei. “Build, Compute, Critique, Repeat: Data Analysis with Latent Variable Models”. In: *Annual Review of Statistics and Its Application* 1.1 (2014), pp. 203–232.
- [7] Dustin Tran et al. “Deep Probabilistic Programming”. In: (Jan. 13, 2017). arXiv: 1701.03757 [cs, stat].
- [8] Bob Carpenter et al. “Stan : A Probabilistic Programming Language”. In: *Journal of Statistical Software* 76.1 (2017).
- [9] John Salvatier, Thomas Wiecki, and Christopher Fonnesbeck. “Probabilistic Programming in Python Using PyMC”. In: (July 29, 2015). arXiv: 1507.08050 [stat].