

# 確率的プログラミング ライブラリ Edward

@ PyConJP 2017



# Yuta Kashino (柏野 雄太)

BakFoo, Inc. CEO

Astro Physics /Observational Cosmology  
Zope / Python



Realtime Data Platform for Enterprise / Prototyping



# Yuta Kashino (柏野 雄太)

仕事, 家事育児, スイム・ランニング,

arXiv読み



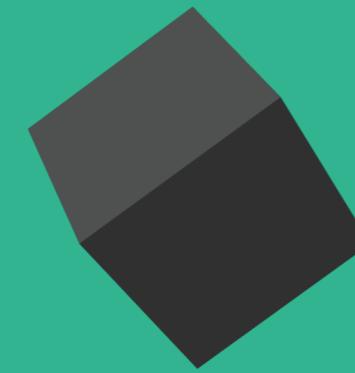
stat.ML, stat.TH, cs.CV, cs.CL, cs.LG

math-ph, astro-ph

PyCon2016 週末サイエンティストのススメ

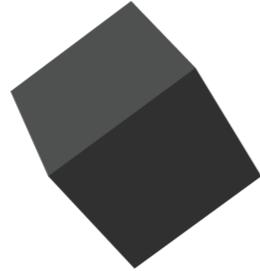
<https://www.slideshare.net/yutakashino/pyconjp2016>

@yutakashino



Edward

# Edward



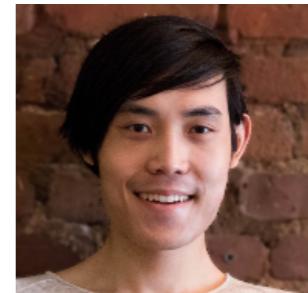
- Dustin Tran (Open AI) がクリエイター
- コロンビア大 Blei Labで生まれ, コア開発者も.
- 確率的プログラミング言語 (PPL)
- Stan, PyMC3, Anglican, Church, Venture, Figaro, WebPPL
- 2016年2月に公開された比較的新しいPPL

- 計算はTensorFlowの肩に乗る



- George **Edward** Pelham Box

Box-Cox Trans., Box-Jenkins, Ljung-Box test, ちなみにbox plotはTukey,  
結婚3回：2番目の嫁はRA Fisherの次女



# Edward



1. TensorFlow 計算グラフ

2. 確率変数

3. 確率モデリング

4. ベイズ推定

PPL

5. Box's loop

TensorFlow(TF) + 確率的プログラミング(PPL)

TF: スケールする計算グラフ

PPL: 確率変数 + 確率モデリング + ベイズ推定

Python/Numpyスタックによるモデリング

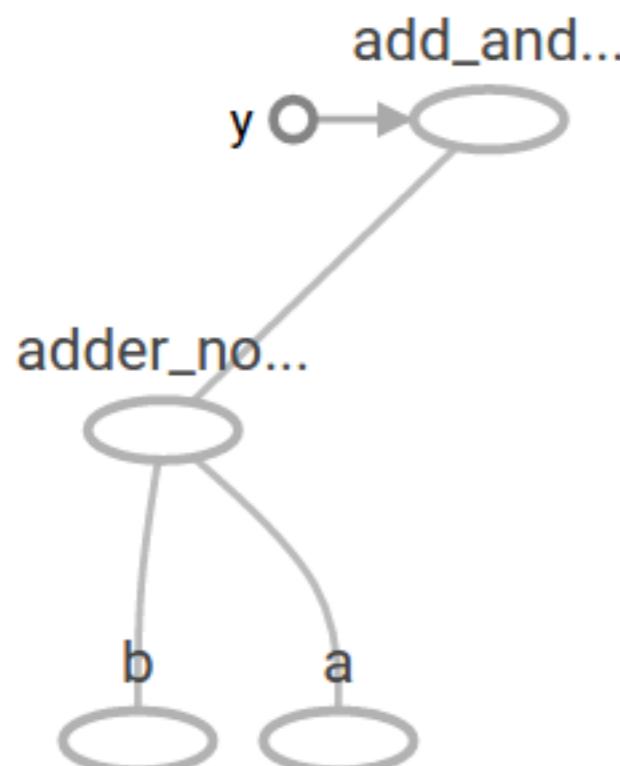


# 1. TF: 計算グラフ

# 1. TF: 計算グラフ



- ノード: 計算操作, ループ
- エッジ: テンソル



```

a = tf.placeholder(tf.float32)
b = tf.placeholder(tf.float32)
adder_node = a + b # + provides a shortcut for tf.add(a, b)

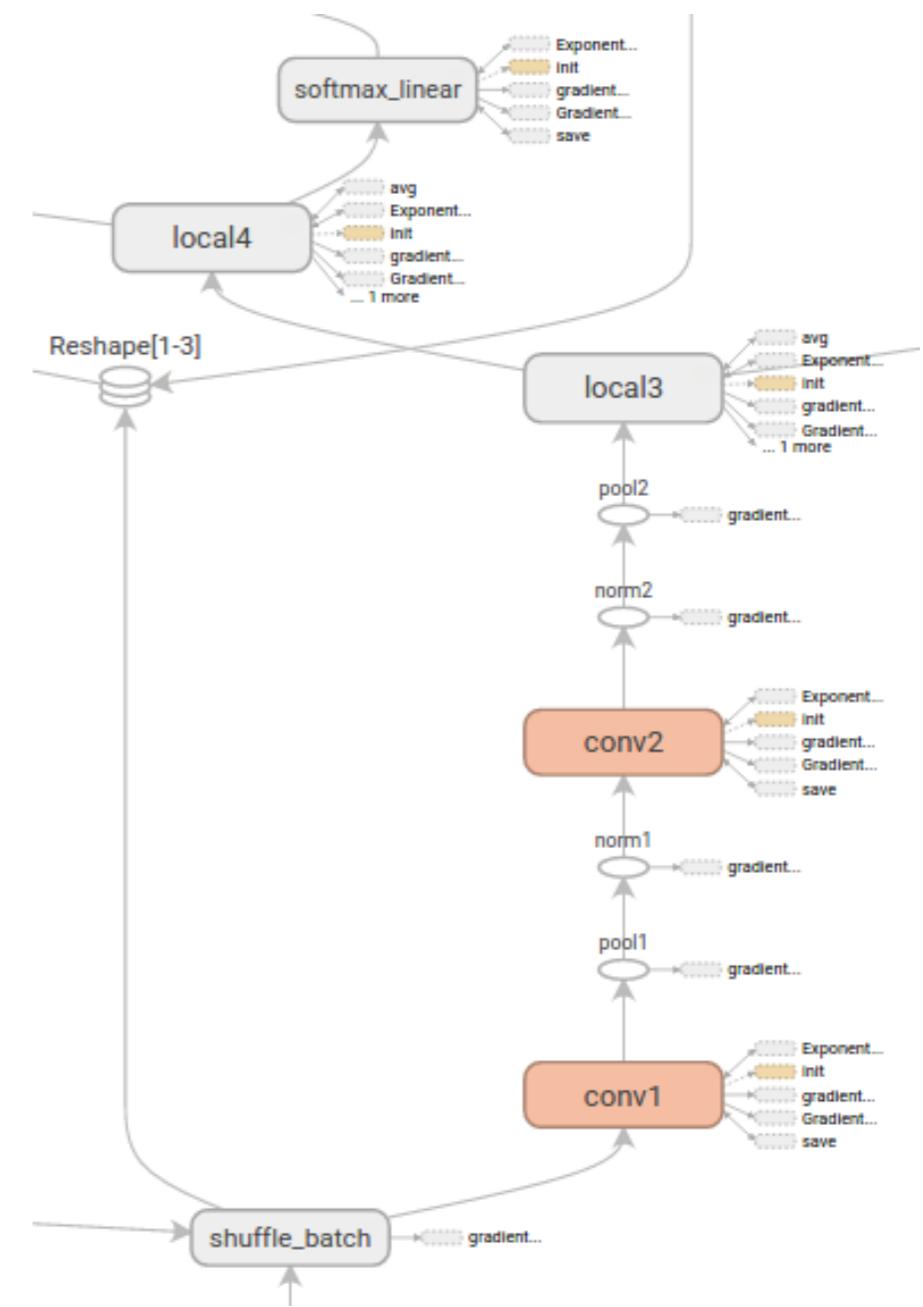
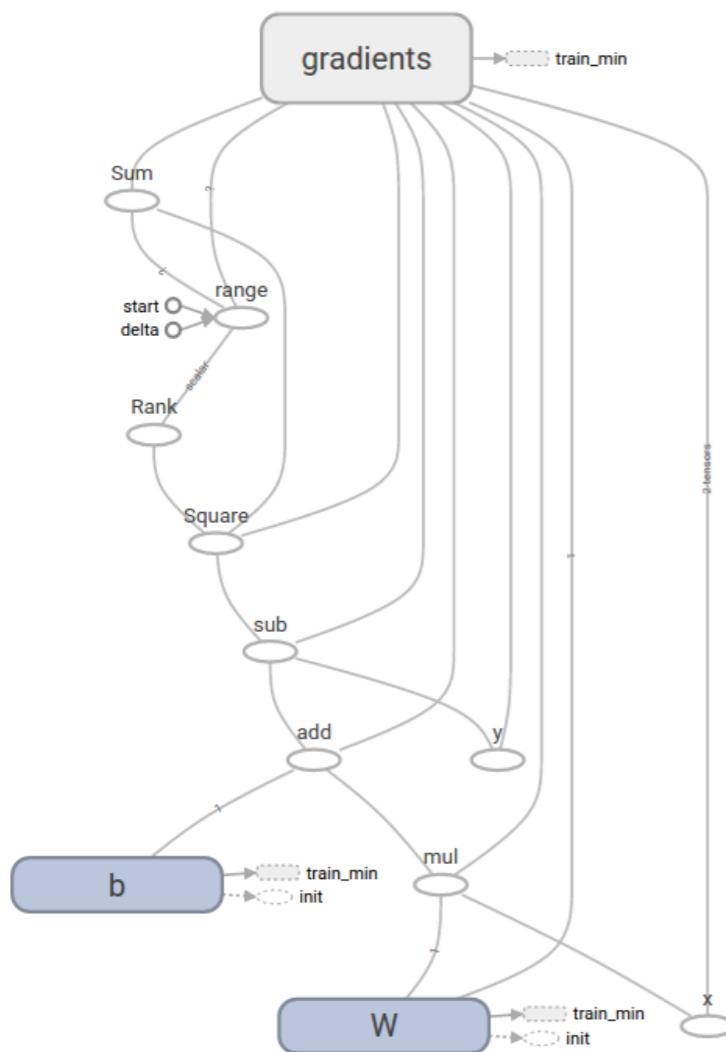
add_and_triple = adder_node * 3.
print(sess.run(add_and_triple, {a: 3, b:4.5}))
  
```

# 1. TF: 計算グラフ



1. TensorFlow 計算グラフ	2. 確率変数	3. 確率モデル	4. ベイズ推定
		5. Box's loop	

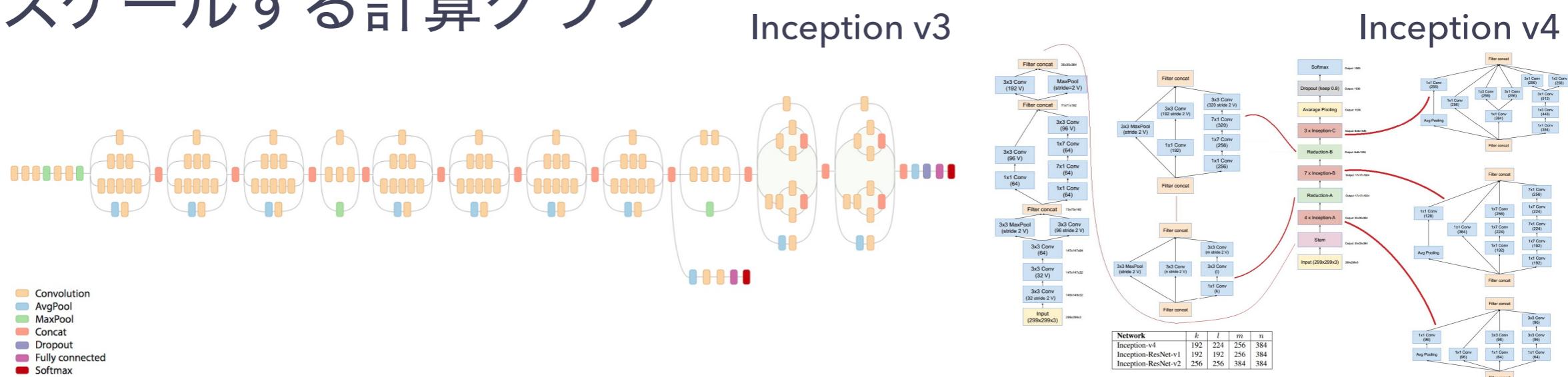
## 深層ニューラルネット



# 1. TF: 計算グラフ

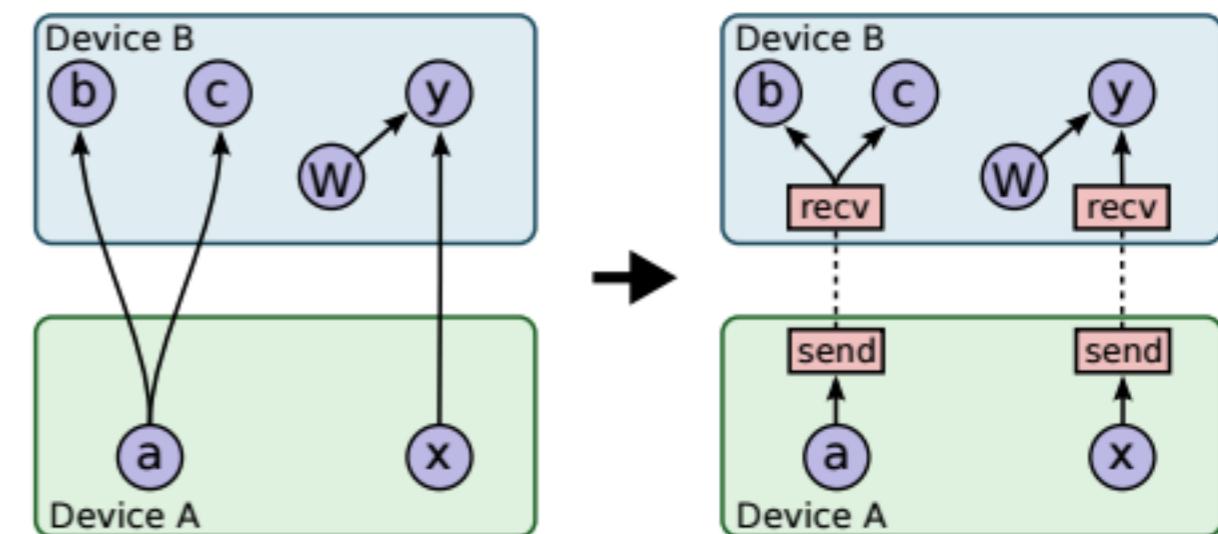
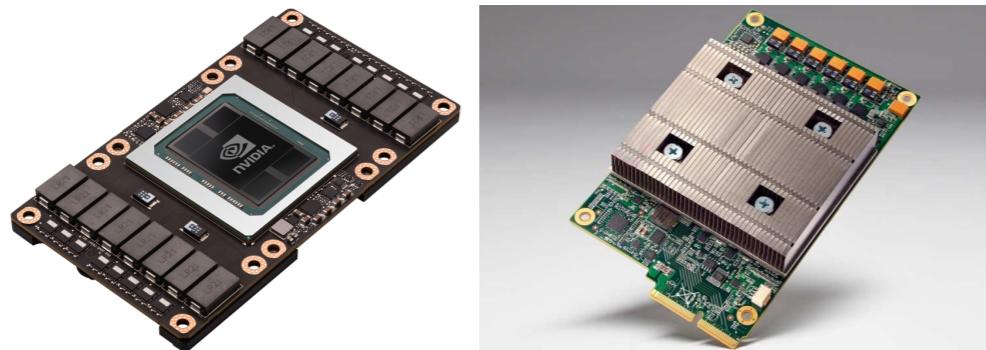


- スケールする計算グラフ



- 計算グラフをデバイス別に分散計算

- GPU / TPU



# 1. TF: 計算グラフ



1. TensorFlow 計算グラフ

2. 確率変数

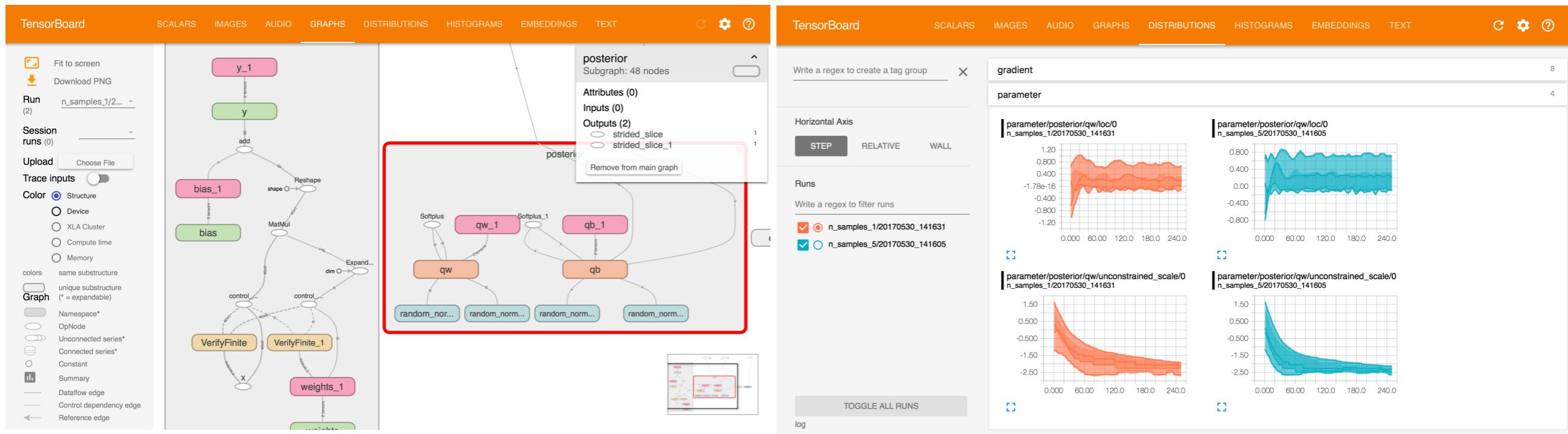
3. 確率モデル

4. ベイズ推定

5. Box's loop

- Keras, Slimが利用できる 

- TensorBoardが利用できる



# 1. TF: 計算グラフ



深層学習で十分でない理由

- 例：DNN: NNのウェイト・バイアスを学習
- 学習済みのウェイト・バイアスは点推定値
- 収束しない問題, 頑健性問題, ハイパーパラメータ問題...
- 点推定値 = 区間推定でもなく確率情報がない

注: DropOutを除く (DropOutはベイズ推定と同等の効果という報告: Yingzhen Li, Yarin Gal ICML, 2017 )

# 1. TF: 計算グラフ



結局のところ：

- 世の中の事象はすべからく確率事象である
  - = できるだけ確率的な情報を保持したい
- TFのテンソルを確率変数に
- 確率変数を取り使うベイズ推定アルゴリズムを導入

だからEdward

## 2. 確率変数

## 2. 確率変数

$x$ : 確率変数, ランダムな試行により得られる結果

$$\mathbf{x}^* \sim P(\mathbf{x} | \alpha)$$

確率分布関数をともなう. 例えばベータ分布の場合

$$\theta^* \sim \text{Beta}(\theta | 1, 1)$$

edwardでは

```
from edward.models import Beta
theta = Beta(1.0, 1.0)
with tf.Session() as sess:
    print(sess.run(theta))
```

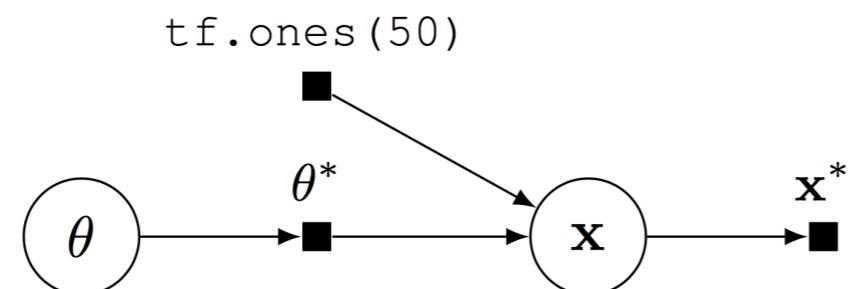
0.323213

## 2. 確率変数

ベータ-ベルヌーイモデル(コイン投げ)

$$p(\mathbf{x}, \theta) = \text{Beta}(\theta | 1, 1) \prod_{n=1}^{50} \text{Bernoulli}(x_n | \theta),$$

グラフィカルモデル



Edwardだと

```

from edward.models import Bernoulli
theta = Beta(1.0, 1.0)
x = Bernoulli(tf.ones(50) * theta)
with tf.Session() as sess:
    print(sess.run(x))
    
```

```

[1 1 0 0 1 0 1 1 1 1 0 1 1 0 1 0 1 1 1 1 0 1 1 1 1 0 0 1 1 1 0 1 1 0 1 1 0
 1 0 1 1 1 1 1 1 0 1 1 1]
    
```

## 2. 確率変数

基本的なツールは一揃いある

- 対数確率  $\log p(\mathbf{x} \mid \theta^*)$

`log_prob()`

- 期待値  $\mathbb{E}_{p(\mathbf{x} \mid \theta^*)}[\mathbf{x}]$   
`mean()`
- サンプリング  $\mathbf{x}^* \sim p(\mathbf{x} \mid \theta^*)$

`sample()`

### 3. 確率モデリング

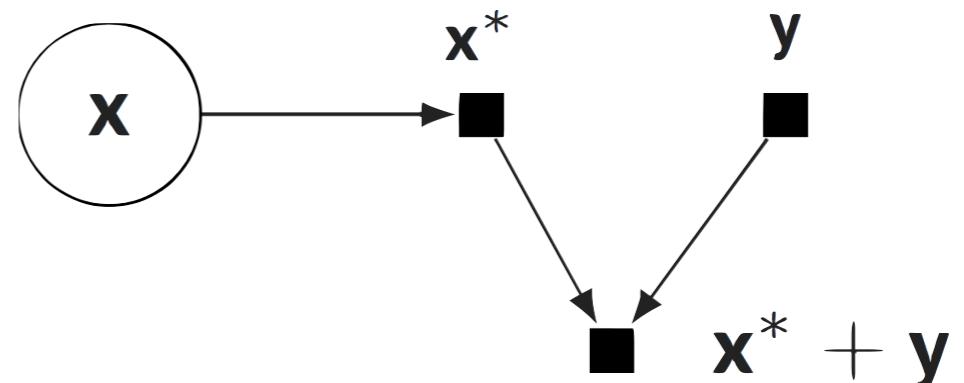
# 3. 確率モデリング

Edwardでは確率モデルをTFの計算グラフとして実現

たとえば以下例

$$x^* \sim \text{Normal}(x | 0, I)$$

$$y \sim \text{constant}$$



```

from edward.models import Normal

x = Normal(loc=tf.zeros(10), scale=tf.ones(10))
y = tf.constant(5.0)
x + y, x - y, x * y, x / y
tf.tanh(x * y)
x[2] # 3rd normal rv in the vector

<tf.Tensor 'strided_slice:0' shape=() dtype=float32>
  
```

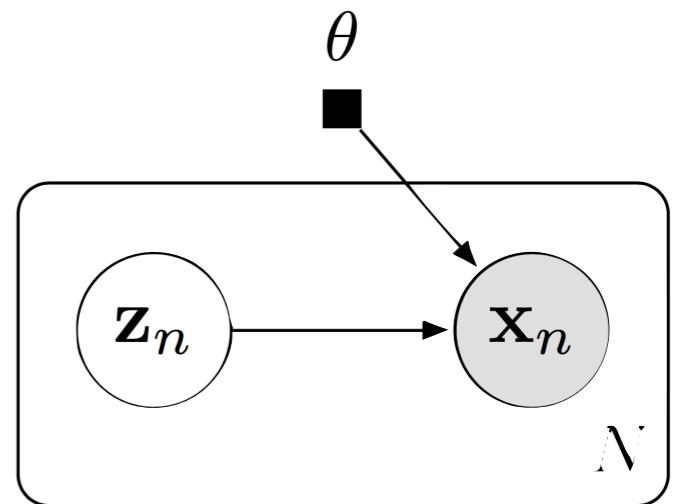
### 3. 確率モデリング

ベイジアンニューラルネットワーク

256の全結合ネット, 出力は $28*28$

$$n = 1 \dots N$$

$$\mathbf{z}_n \sim \text{Normal}(\mathbf{z}_n \mid \mathbf{0}, \mathbf{I}),$$



$$\mathbf{x}_n \mid \mathbf{z}_n \sim \text{Bernoulli}(\mathbf{x}_n \mid p = \text{NN}(\mathbf{z}_n; \theta)).$$

```

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

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

## 4. ベイズ推定

# 4. ベイズ推定

観測データ  $\mathbf{X}$ , 隠れ変数  $\mathbf{Z}$ ,  $\mathbf{Z}$ を事後確率として推定

ベイズの定理

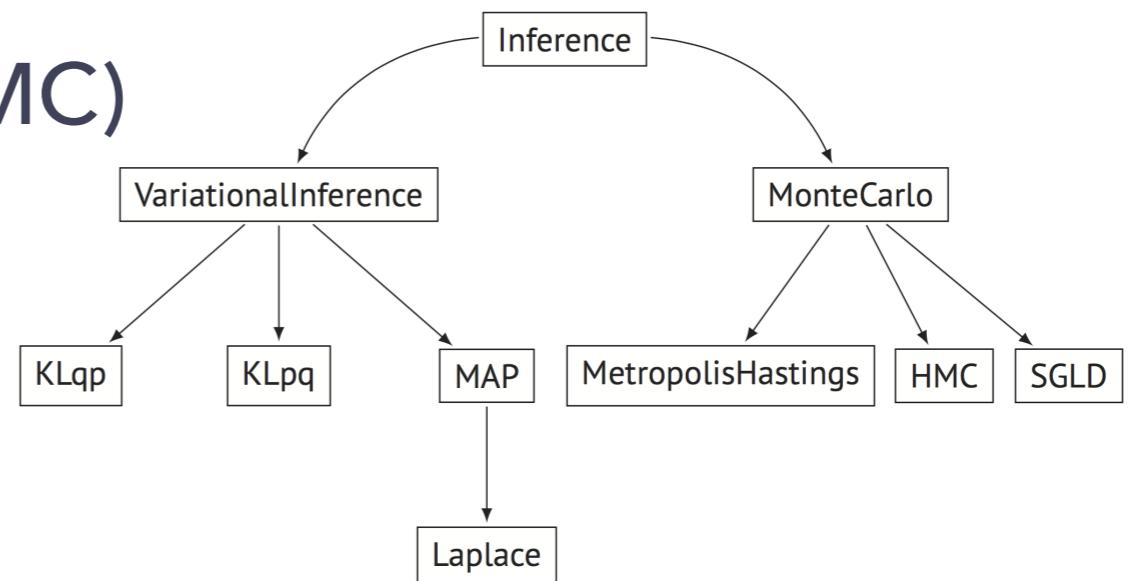
$$p(\mathbf{z} \mid \mathbf{x}) = \frac{p(\mathbf{x}, \mathbf{z})}{\int p(\mathbf{x}, \mathbf{z}) d\mathbf{z}}.$$

積分できない

そこでベイズ推定の近似

- 変分推定 (Variational Bayes)

- マルコフ鎖モンテカルロ (MCMC)



# 4. ベイズ推定

- o `edward.inferences.Inference`

- o `edward.inferences.VariationalInference`

- o `edward.inferences.MonteCarlo`



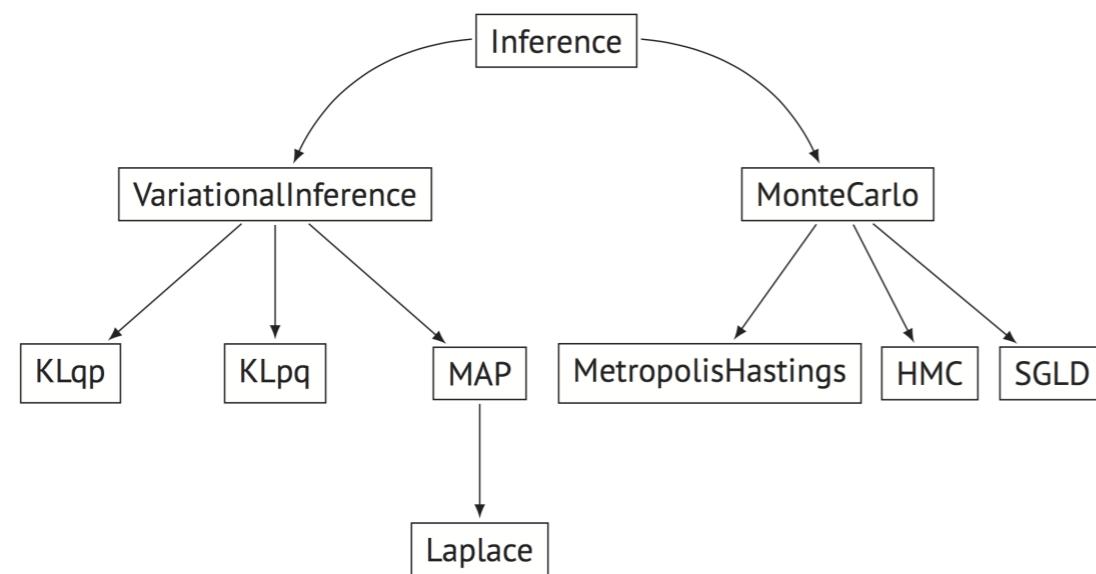
- o `edward.inferences.Gibbs`

- o `edward.inferences.MetropolisHastings`

- o `edward.inferences.HMC`

- o `edward.inferences.SGLD`

- o `edward.inferences.SGHMC`



- o `edward.inferences.KLqp`

- o `edward.inferences.ReparameterizationKLqp`

- o `edward.inferences.ReparameterizationKLKLqp`

- o `edward.inferences.ReparameterizationEntropyKLqp`

- o `edward.inferences.ScoreKLqp`

- o `edward.inferences.ScoreKLKLqp`

- o `edward.inferences.ScoreEntropyKLqp`

- o `edward.inferences.KLpq`

- o `edward.inferences.GANInference`

- o `edward.inferences.BiGANInference`

- o `edward.inferences.ImplicitKLqp`

- o `edward.inferences.WGANInference`

- o `edward.inferences.MAP`

- o `edward.inferences.Laplace`

# 4. ベイズ推定

## 変分推定

$p(\mathbf{z}|\mathbf{x})$ を直接求めない。KL距離を最小にする $q(\mathbf{z})$

$$\begin{aligned}\lambda^* &= \arg \min_{\lambda} \text{KL}(q(\mathbf{z} ; \lambda) \| p(\mathbf{z} | \mathbf{x})) \\ &= \arg \min_{\lambda} \mathbb{E}_{q(\mathbf{z} ; \lambda)} [\log q(\mathbf{z} ; \lambda) - \log p(\mathbf{z} | \mathbf{x})].\end{aligned}$$

しかしこれもムリなので、ELBOを最大化する。

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

$$\lambda^* = \arg \max_{\lambda} \text{ELBO}(\lambda).$$

# 4. ベイズ推定

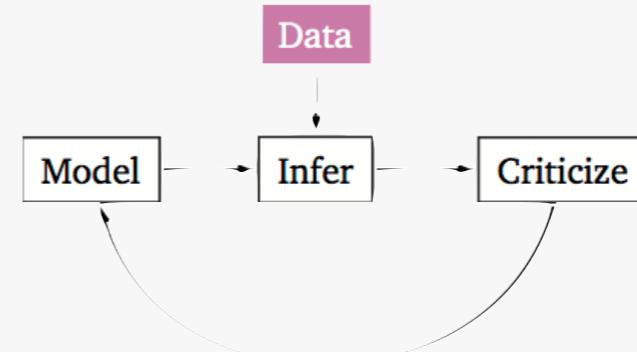
Edwardでの変分推定KLqp最小化は以下

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

# Model
p = Beta(1.0, 1.0)
x = Bernoulli(tf.ones(10) * p)

# Inference
qp_a = tf.nn.softplus(tf.Variable(0.0))
qp_b = tf.nn.softplus(tf.Variable(0.0))
qp = Beta(qp_a, qp_b)

inference = ed.KLqp({p: qp}, {x: x_data})
inference.run(n_iter=500)
```



500/500 [100%] ███ Elapsed: 0s | Loss: nan

# 5. Box's loop

# 5. Box's loop

George Edward Pelham Box



1. TensorFlow 計算グラフ

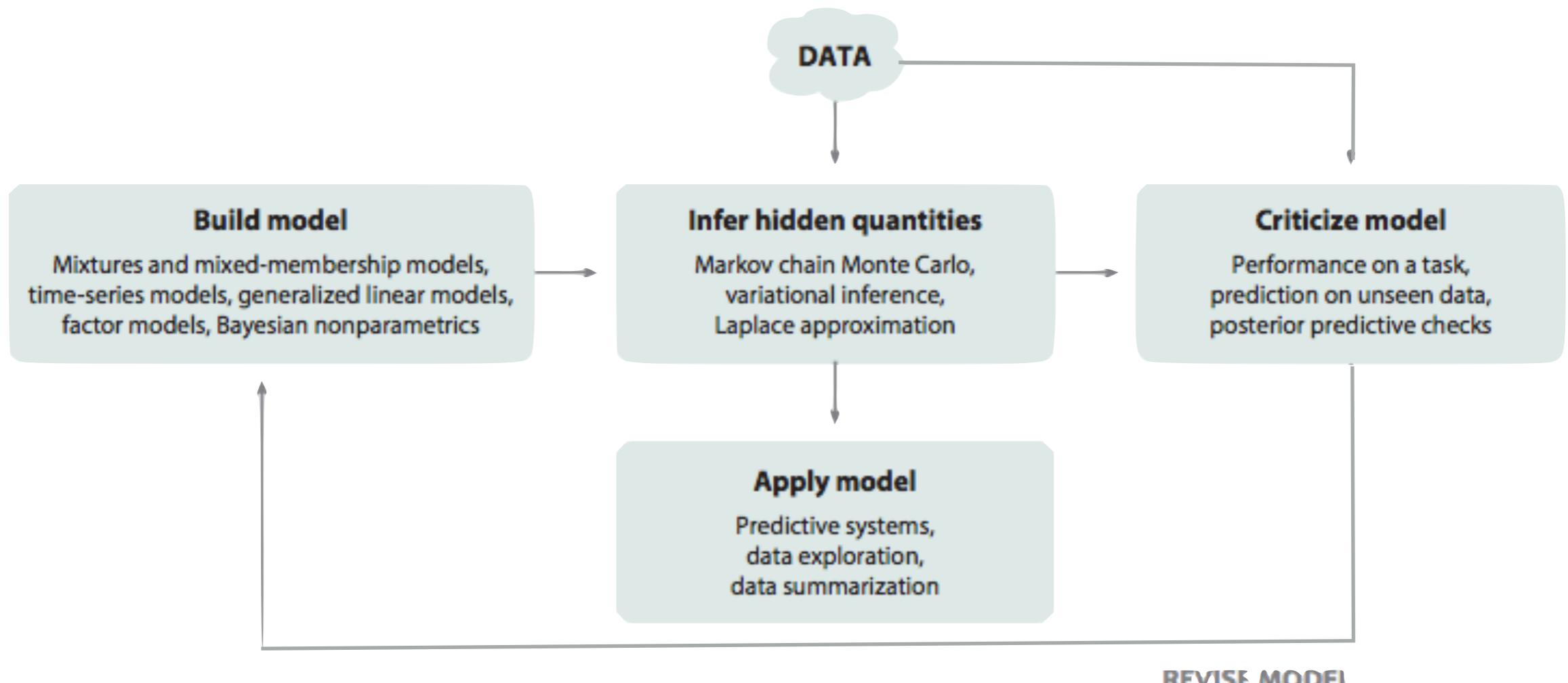
2. 確率変数

3. 確率モデル

4. ベイズ推定

5. Box's loop

Blei 2014



# 5. Box's loop

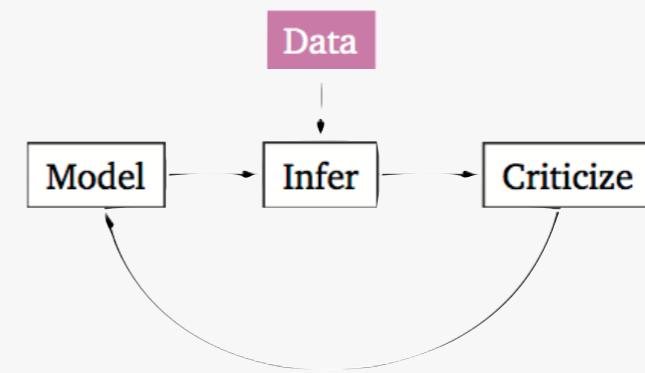
```
# Data
x_data = np.array([0, 1, 0, 0, 0, 0, 0, 0, 0, 1])

# Model
p = Beta(1.0, 1.0)
x = Bernoulli(tf.ones(10) * p)

# Inference
qp_a = tf.nn.softplus(tf.Variable(0.0))
qp_b = tf.nn.softplus(tf.Variable(0.0))
qp = Beta(qp_a, qp_b)

inference = ed.KLqp({p: qp}, {x: x_data})
inference.run(n_iter=500)
```

500/500 [100%] ██████████████████████ Elapsed: 0s | Loss: nan



```
# Criticism

# build posterior predictive after inference:
# it is parameterized by a posterior sample
x_post = ed.copy(x, {p: qp})

# posterior predictive check
# T is a user-defined function of data, T(data)
# in general T is a discrepancy function of the data (both response and
# covariates) and latent variables, T(data, latent_vars)
T = lambda xs, zs: tf.reduce_mean(xs[x_post])

# prior predictive check
# run ppc on original x
ppc1 = ed.ppc(T, data={x_post: x_data})
```

# Edwardまとめ

- Edward = TensorFlow + 確率変数 + ベイズ推定
- TensorFlowという巨人の肩にのる
- スケールする計算グラフ
- TFエコシステム：GPU, TPU, TensorBoard, Keras
- 確率変数, ベイズ推定アルゴリズムのセット
- Box's Loop
- Pythonベースで習得が容易

# おまけ

## Experiment: 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)
```

We perform inference on Bayesian logistic regression for the Covertype dataset ( $N = 581012$ ,  $D = 54$ ). We use a 12-core Intel i7-5930K CPU at 3.50GHz and a NVIDIA Titan X (Maxwell) GPU.

We compare the runtime of HMC for  $T = 100$  iterations (and same settings).

---

Probabilistic programming language	Runtime
Stan (1 CPU) [2]	171 sec
PyMC3 (12 CPU) [6]	361 sec
<b>Edward (12 CPU)</b>	<b>8.2 sec</b>
<b>Edward (GPU)</b>	<b>4.9 sec (35x faster than Stan)</b>

---

# Refrence

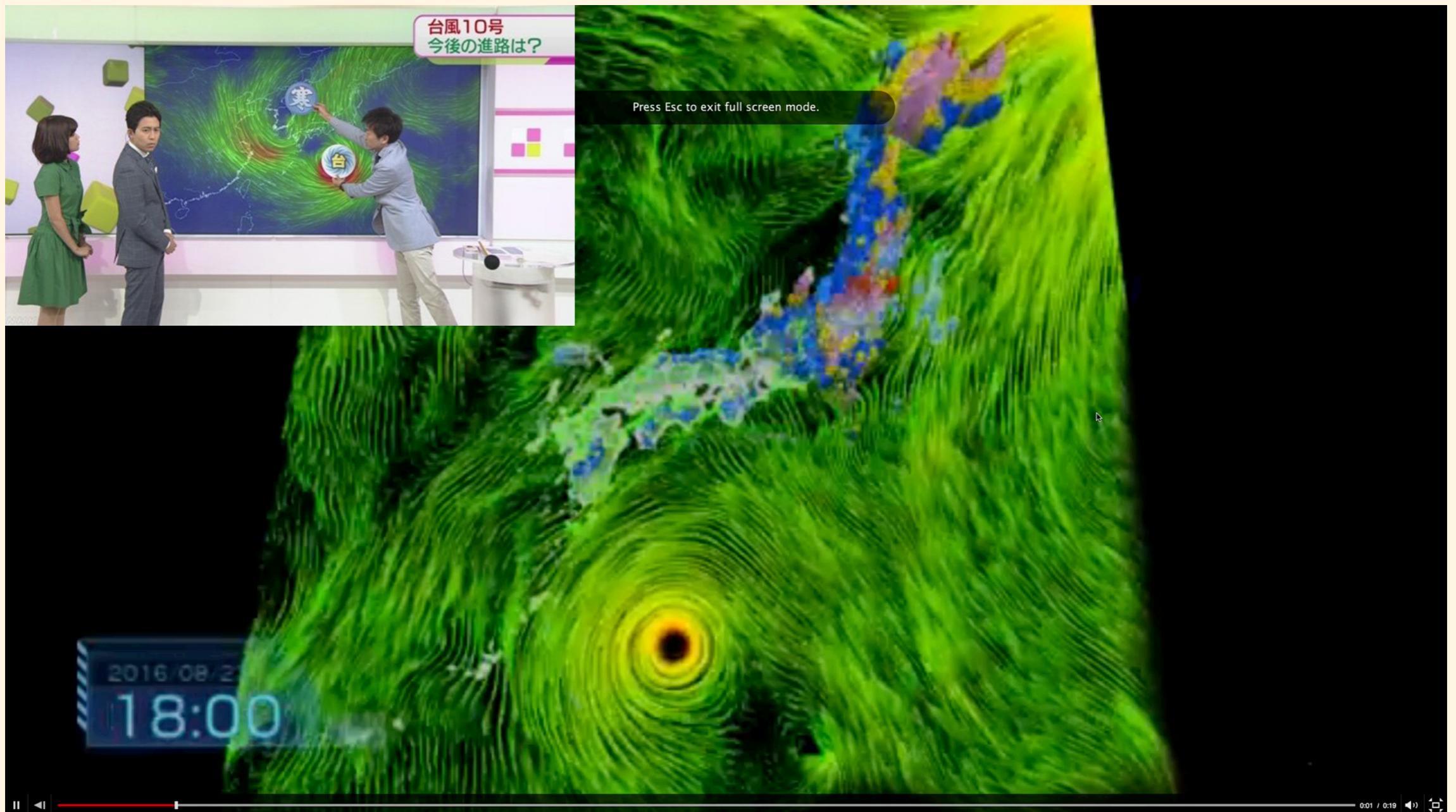
- D. Tran, A. Kucukelbir, A. Dieng, M. Rudolph, D. Liang, and D.M. Blei. Edward: A library for probabilistic modeling, inference, and criticism.(arXiv preprint arXiv:1610.09787)
- D. Tran, M.D. Hoffman, R.A. Saurous, E. Brevdo, K. Murphy, and D.M. Blei. Deep probabilistic programming.(arXiv preprint arXiv:1701.03757)
- Box, G. E. (1976). Science and statistics. (*Journal of the American Statistical Association*, 71(356), 791-799.)
- D.M. Blei. Build, Compute, Critique, Repeat: Data Analysis with Latent Variable Models. (*Annual Review of Statistics and Its Application* Volume 1, 2014)

# Questions

[kashino@bakfoo.com](mailto:kashino@bakfoo.com)  
[@yutakashino](https://twitter.com/yutakashino)

# BakFoo, Inc.

NHK NMAPS: リアルタイムデータ + 可視化



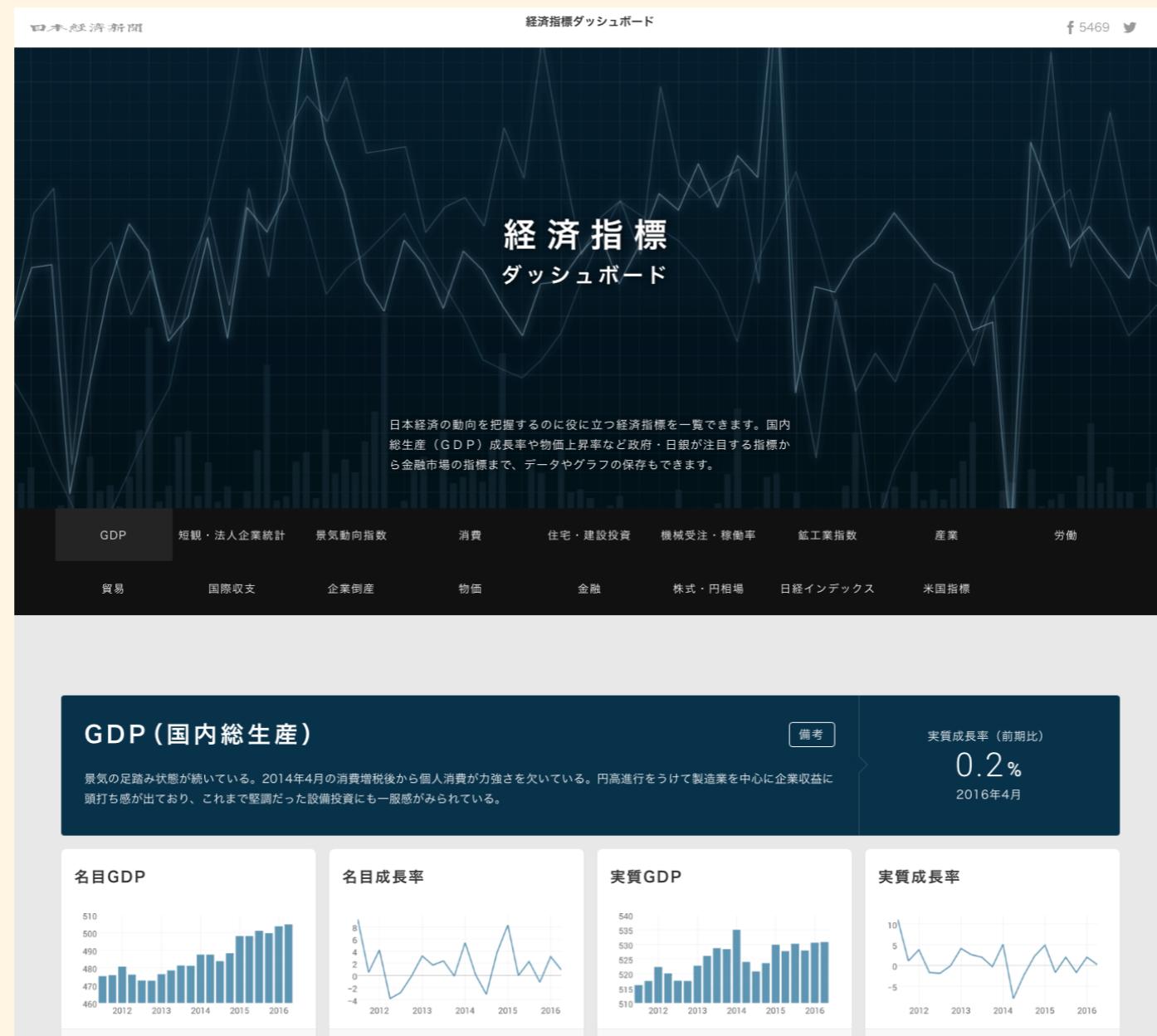
# オープンデータ+マイクロサービス



PyConJP 2015  
日本のオープンデータ  
プラットフォーム  
をPythonでつくる

# BakFoo, Inc.

## 日本経済新聞社 経済指標ダッシュボード



# BakFoo, Inc.

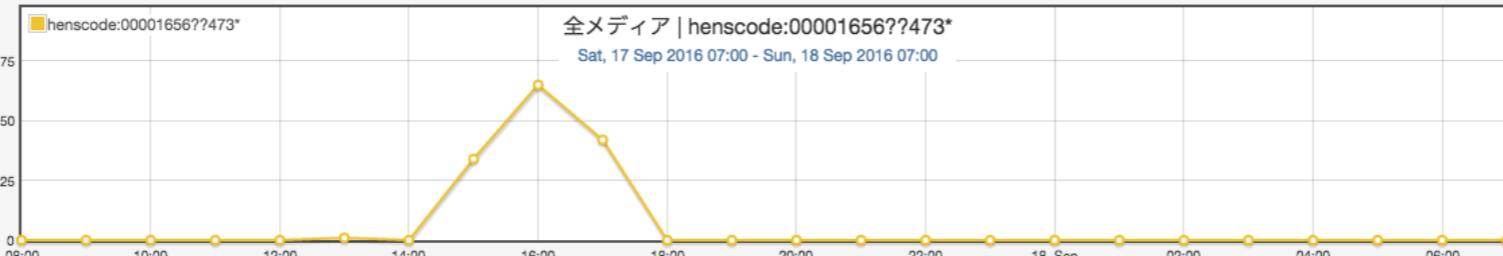
## テレビ番組判定: SNS + 自然言語解析

編成コード検索: 1656473 | 472 | 474

番組情報		Tweet Counts	Shared Counts
番組 大相撲秋場所 七日目 ▽ゲスト 原沢久喜選手（リオ五輪 柔道 銀メダリスト）	OnAir 106	Twitter 106	
概要 ▽ゲスト 原沢久喜選手（リオ五輪 柔道 100キロ超級 銀メダリスト）（4:10）「幕内取組」【ゲスト】原沢久喜、【解説】正面（幕内）舞の海秀平	Total 145	facebook 5	
放送時間 2016年09月17日 16:00:00 ~ 2016年09月17日 18:00:00	Avg/min 0.87	hatebu 0	
	Max/min 6	PC NA 0 5	
		Mobile NA 0 0	
		OnDemand NA NA NA	

Keywords	
大相撲秋場所 88 , sumo 74 , 原沢久喜 51 , ゲスト 50 , リオ五輪 49 , 銀メダリスト 49 , 中の人 46 , リプライ 46 , 幕内取組 34 , アナウンサー 30 , 大坂敬久 28 , 舞の海秀平 28 , nhk総合 20 , ツイート 12 , 総レス 12 , アナログラジオ 7 , 大相撲 7 , 今日は 7 , 西岩親方 6 , 若の里 6 , 舞の海 5 , 大相撲中継 5 , 豊真将 4 , 手拍子 4 , ラジオ 4 , 松鳳山 3 , 激アツ 3 , 盛り上がり状態 3 , 千代の国 3 , 朝青龍 3 , 隠岐の海 3 , 向正面 3 , 戸部眞輔 2 , 春日野 2 , 固形シャンプー 2 , 日馬富士 2 , 決まり手 2 , シャンプー 2 , リオオリンピック 2 , 伊勢ヶ濱部屋 2	Twitter 106

全メディア | henscode:00001656??473\*  
Sat, 17 Sep 2016 07:00 - Sun, 18 Sep 2016 07:00



舞の海さんリメールの話しかしないな~(笑) #sumo #nhk  
2016年09月17日 17:35:02  
かなぶん  
@knzwysnr OA

HOT実況 【137人が実況中】大相撲秋場所 七日目 ▽ゲスト 原沢久喜選手（リオ五輪 柔道 銀メダリスト）【勢い:37tw/分】livetter.com/tv/ch1/ 2016年09月17日 17:34:01  
#nhk #sumo HOTテレビ実況なう☆Livetter @livetter\_hot OA

「稀に勢いの出る大関」の「稀」の時期かな #sumo #nhk 2016年09月17日 17:32:32  
きみすたすとりーと@沼津より愛を込めて @sazankuwata OA

お稀勢さん勝ちました。 #nhk #sumo 2016年09月17日 17:32:21  
かすみ猫 @ich\_kasumi OA

万全ではないが勝った #NHK #sumo 2016年09月17日 17:32:20  
たにま @tanimax\_GoS OA