Computing lp__

proportionality constants and (automatic) transforms

Daniel Lee

daniel@generable.com

May 22, 2018. Stan Meetup.

Generable

Who am I?

- Stan developer since 2011

- CTO of Generable

  Tools in pharma for analysis of early stage clinical trials

What is Stan?

1. Statistical Modeling Language

2. Inference algorithms

3. Interfaces

What is Stan?

1. Statistical Modeling Language

   - purpose: specify data, parameters, log joint probability distribution

$$\theta,\ x,\ \log p(\theta, x)$$

2. Inference algorithms

3. Interfaces

What is Stan?

1. Statistical Modeling Language

   - purpose: specify data, parameters, log joint probability distribution

$$\theta,\ x, \boxed{\log p(\theta, x)}$$

This is a function of theta

2. Inference algorithms

3. Interfaces

What is Stan?

1. Statistical Modeling Language

   - purpose: specify data, parameters, log joint probability distribution

$$\theta,\ x,\ \log p(\theta, x)$$

2. Inference algorithms

   - Bayesian (NUTS, HMC), approximate Bayes (ADVI), optimization (L-BFGS, BFGS, Newton)

$$p(\theta \mid x)$$

with

$$\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(N)}$$

3. Interfaces

What is Stan?

1. Statistical Modeling Language

   - purpose: specify data, parameters, log joint probability distribution

$$\theta,\ x,\ \log p(\theta, x)$$

2. Inference algorithms

   - Bayesian (NUTS, HMC), approximate Bayes (ADVI), optimization (L-BFGS, BFGS, Newton)

$$\hat{p}(\theta \mid x) \approx q(\hat{\phi})$$

where

$$\hat{\phi} = \underset{\phi}{\operatorname{argmin}}\ D_{\mathrm{KL}}\left(q(\theta \mid \phi) \,||\, p(\theta \mid x)\right)$$

3. Interfaces

What is Stan?

1. Statistical Modeling Language

   - purpose: specify data, parameters, log joint probability distribution

$$\theta,\ x,\ \log p(\theta, x)$$

2. Inference algorithms

   - Bayesian (NUTS, HMC), approximate Bayes (ADVI), optimization (L-BFGS, BFGS, Newton)

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}}\, p(\theta, x)$$

3. Interfaces

What is Stan?

1. Statistical Modeling Language

   - purpose: specify data, parameters, log joint probability distribution

$$\theta,\ x,\ \log p(\theta, x)$$

2. Inference algorithms

   - Bayesian (NUTS, HMC), approximate Bayes (ADVI), optimization (L-BFGS, BFGS, Newton)

$$p(\theta \mid x)$$

with

$$\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(N)}$$

$$\hat{p}(\theta \mid x) \approx q(\hat{\phi})$$

where

$$\hat{\phi} = \operatorname*{argmin}_{\phi}\ D_{\mathrm{KL}}\left(q(\theta \mid \phi) \,||\, p(\theta \mid x)\right)$$

$$\hat{\theta} = \operatorname*{argmax}_{\theta} p(\theta, x)$$

3. Interfaces

What is Stan?

1. Statistical Modeling Language

   - purpose: specify data, parameters, log joint probability distribution

$$\theta,\ x,\ \log p(\theta, x)$$

2. Inference algorithms

   - Bayesian (NUTS, HMC), approximate Bayes (ADVI), optimization (L-BFGS, BFGS, Newton)

$$p(\theta \mid x)$$

with

$$\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(N)}$$

$$\hat{p}(\theta \mid x) \approx q(\hat{\phi})$$

where

$$\hat{\phi} = \underset{\phi}{\operatorname{argmin}}\ D_{\mathrm{KL}}\left(q(\theta \mid \phi) \,||\, p(\theta \mid x)\right)$$

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}}\, p(\theta, x)$$

3. Interfaces

   - CmdStan, RStan, PyStan; stan.jl, StataStan, MathematicaStan, MatlabStan, ...

What is lp__

- Reported at every iteration

```
lp__,accept_stat__,stepsize__,treedepth__,n_leapfrog__,divergent__,energy__,theta
-7.31587,0.937769,0.975509,1,1,0,7.38337,0.395893
-7.31587,0.639067,0.975509,1,1,0,8.31397,0.395893
-7.31587,0.796836,0.975509,1,1,0,7.8854,0.395893
-6.89987,0.823618,0.975509,2,3,0,8.32733,0.322701
-7.31106,0.745771,0.975509,2,3,0,8.41781,0.395235
-6.7669,1,0.975509,1,3,0,7.19775,0.274794
-6.78954,0.955721,0.975509,2,3,0,7.0435,0.215185
-6.8603,0.9827,0.975509,2,3,0,6.94781,0.1941
```

What is lp__

- It's not just $\log\ p(\theta, x)$

- Evaluation of the model on the **unconstrained scale**
- what's "constrained" and "unconstrained"?

- Full details are in the manual

35. Transformations of Constrained Variables

To avoid having to deal with constraints while simulating the Hamiltonian dynamics during sampling, every (multivariate) parameter in a Stan model is transformed to an unconstrained variable behind the scenes by the model compiler. The transform is based on the constraints, if any, in the parameter's definition. Scalars or the scalar values in vectors, row vectors or matrices may be constrained with lower and/or upper bounds. Vectors may alternatively be constrained to be ordered, positive ordered, or simplexes. Matrices may be constrained to be correlation matrices or covariance matrices. This chapter provides a definition of the transforms used for each type of variable.

Stan converts models to C++ classes which define probability functions with support on all of $\mathbb{R}^K$, where K is the number of unconstrained parameters needed to define the constrained parameters defined in the program. The C++ classes also include code to transform the parameters from unconstrained to constrained and apply the appropriate Jacobians.

Important math

- Bayes rule:

$$p(\theta \mid x) = \frac{p(\theta, x)}{p(x)}$$

Important math

- Bayes rule:

$$p(\theta \mid x) = \frac{p(\theta, x)}{p(x)}$$

$$p(\theta \mid x) \propto p(\theta, x)$$

Important math

- Bayes rule:

$$p(\theta \mid x) = \frac{p(\theta, x)}{p(x)}$$

$$\boxed{p(\theta \mid x) \propto p(\theta, x)}$$

What does this mean???

Important math

- Bayes rule:

$$p(\theta \mid x) = \frac{p(\theta, x)}{p(x)}$$

$$p(\theta \mid x) \propto p(\theta, x)$$

- This means:

$$p(\theta \mid x) = p(\theta, x) \times g(x)$$

$$\log\, p(\theta \mid x) = \log\, p(\theta, x) + \log g(x)$$

Important math

- Bayes rule:

$$p(\theta \mid x) = \frac{p(\theta, x)}{p(x)}$$

$$p(\theta \mid x) \propto p(\theta, x)$$

- This means:

$$p(\theta \mid x) = p(\theta, x) \times g(x)$$

$$\log\ p(\theta \mid x) = \log\ p(\theta, x) + \log g(x)$$

- NUTS, HMC, or MH needs:

$$f(\theta) = \log\ p(\theta, x) + C$$

Example: constrained = unconstrained

example-1.stan

```
transformed data {
  vector[7] y = [-3, -2, -1, 0, 1, 2, 3]';
}
parameters {
  real mu;
}
model {
  target += normal_lpdf(y | mu, 1);
}
```

$$\text{Normal}(y|\mu,\sigma) = \frac{1}{\sqrt{2\pi}\,\sigma}\exp\left(-\frac{1}{2}\left(\frac{y-\mu}{\sigma}\right)^2\right)$$

$$\text{normal\_lpdf}(y\,|\,\mu,\sigma) = \log(1) - \log\left(\sqrt{(2\pi)}\sigma\right) - \frac{1}{2}\left(\frac{y-\mu}{\sigma}\right)^2$$

Example: constrained = unconstrained

- What's the value of lp__ for mu = 0?
- What's the value of lp__ for mu = 1?

$$\text{normal\_lpdf}(y \mid \mu, \sigma) = \log(1) - \log\left(\sqrt{(2\pi)}\sigma\right) - \frac{1}{2}\left(\frac{y-\mu}{\sigma}\right)^2$$

Example: constrained = unconstrained

- What's the value of lp__ for mu = 0?
- What's the value of lp__ for mu = 1?
- R Code:

```
dnorm(-3:3, 0, 1)
prod(dnorm(-3:3, 0, 1))
log(prod(dnorm(-3:3, 0, 1)))
sum(dnorm(-3:3, 0, 1, log = TRUE))
```

- `lp__ with mu = 0: -20.43257`
- `lp__ with mu = 1: -23.93257`

EXTRA CREDIT: calc diff between two evals

Example: constrained = unconstrained

- Run the model!
- ./example-1 sample output file=output-1.csv

```
lp__,accept_stat__,stepsize__,treedepth__,n_leapfrog__,divergent__,energy__,mu
-20.4658,1,1.07003,2,3,0,20.5726,-0.0974355
-20.4326,0.70331,1.07003,2,3,0,22.69,-0.00199139
```

- Look at histogram of mu!
- Make sense?

Example: drop constants

example-2.stan

```
transformed data {
  vector[7] y = [-3, -2, -1, 0, 1, 2, 3]';
}
parameters {
  real mu;
}
model {
  y ~ normal(mu, 1);
}
```

What constants can be dropped?

$$\text{normal\_lpdf}(y \mid \mu, \sigma) = \log(1) - \log\left(\sqrt{(2\pi)}\sigma\right) - \frac{1}{2}\left(\frac{y-\mu}{\sigma}\right)^2$$

Example: drop constants

example-2.stan

```
transformed data {
  vector[7] y = [-3, -2, -1, 0, 1, 2, 3]';
}
parameters {
  real mu;
}
model {
  y ~ normal(mu, 1);
}
```

since sigma is constant

$$\text{normal\_lpdf}(y \mid \mu, \sigma) = \log(1) - \log\left(\sqrt{(2\pi)}\sigma\right) - \frac{1}{2}\left(\frac{y-\mu}{\sigma}\right)^2$$

$$= -\frac{1}{2}\left(\frac{y-\mu}{1}\right)^2$$

Example: drop constants

- What's the value of lp__ for mu = 0?
- What's the value of lp__ for mu = 1?

$$\text{normal\_lpdf}(y \mid \mu, \sigma) = \log(1) - \log\left(\sqrt{(2\pi)}\sigma\right) - \frac{1}{2}\left(\frac{y-\mu}{\sigma}\right)^2$$

$$= -\frac{1}{2}\left(\frac{y-\mu}{1}\right)^2$$

Example: drop constants

- What's the value of lp__ for mu = 0?
- What's the value of lp__ for mu = 1?

- In R:

```
f <- function(y, mu) {
  return(-0.5 * (y - mu)^2)
}
sum(f(-3:3, 0))
sum(f(-3:3, 1))
```

- `lp__ with mu = 0: -14`
- `lp__ with mu = 1: -17.5`

EXTRA CREDIT: calc diff between two evals

Example: drop constants

- Run the model!
- ./example-2 sample output file=output-2.csv

```
lp__,accept_stat__,stepsize__,treedepth__,n_leapfrog__,divergent__,energy__,mu
-14.0074,0.876665,1.14317,2,3,0,14.6728,0.045892
-14.0503,0.984525,1.14317,1,1,0,14.0507,0.119911
-14.1077,0.825536,1.14317,2,3,0,14.6893,0.175427
```

- Look at histogram of mu!
- Make sense?

Example: constants don't matter!!!!

example-3.stan

```
transformed data {
  vector[7] y = [-3, -2, -1, 0, 1, 2, 3]';
}
parameters {
  real mu;
}
model {
  y ~ normal(mu, 1);
  target += 12345;
  target += mean(exp(y));
}
```

Example: constants don't matter!!!

- Run the model!
- ./example-3 sample output file=output-3.csv

```
lp__,accept_stat__,stepsize__,treedepth__,n_leapfrog__,divergent__,energy__,mu
-20.4658,1,1.07003,2,3,0,20.5726,-0.0974355
-20.4326,0.70331,1.07003,2,3,0,22.69,-0.00199139
```

- Look at histogram of mu!
- Make sense?

Transformations: see ch 35 for more info

- Transform:
  - one-to-one function
  - takes constraints on variables (lower = 0, simplex, covariance matrix) and makes it unconstrained (range: -inf to inf)
  - Univariate:

$$p_Y(y) = p_X(f^{-1}(y)) \left| \frac{d}{dy} f^{-1}(y) \right|$$

Transformations: see ch 35 for more info

- Transform:
  - one-to-one function
  - takes constraints on variables (lower = 0, simplex, covariance matrix) and makes it unconstrained (range: -inf to inf)
  - Univariate:

$$p_Y(y) = p_X(f^{-1}(y)) \boxed{\left| \frac{d}{dy} f^{-1}(y) \right|}$$

abs derivative of inverse transform

Transformations: see ch 35 for more info

- In general, multivariate

$$p_Y(y) = p_X(f^{-1}(y)) \boxed{\left| \det J_{f^{-1}}(y) \right|}$$

abs determinant of Jacobian

$$J_{f^{-1}}(y) = \begin{bmatrix} \frac{\partial x_1}{\partial y_1} & \cdots & \frac{\partial x_1}{\partial y_K} \\ \vdots & \vdots & \vdots \\ \frac{\partial x_K}{\partial y_1} & \cdots & \frac{\partial x_K}{\partial y_K} \end{bmatrix}$$

Example: automatic transform

example-4.stan

```
parameters {
  real<lower = 0, upper = 1> theta;
}
model {
}
```

Why is lp__ not exactly 0?

Example: automatic transform

```
real<lower = 0, upper = 1> theta;
```

- Constrained variable between 0 and 1
- Unconstrained between -inf and inf
- $v \in (-\infty, \infty)$ $\qquad \text{logit}^{-1}(v) = \dfrac{1}{1 + \exp(-v)}$
- $$\frac{d}{dy}\text{logit}^{-1}(y) = \text{logit}^{-1}(y) \cdot \left(1 - \text{logit}^{-1}(y)\right)$$

Example: automatic transform

```
real<lower = 0, upper = 1> theta;
```

$$p_Y(y) = p_X\big(a + (b-a)\cdot \text{logit}^{-1}(y)\big)\cdot(b-a)\cdot \text{logit}^{-1}(y)\cdot\big(1-\text{logit}^{-1}(y)\big)$$

- `a = 0, b = 1`
- `For our model:` $\text{logit}^{-1}(y) = \theta$

Example: automatic transform

```
real<lower = 0, upper = 1> theta;
```

$$p_Y(y) = p_X\big(a + (b-a)\cdot \text{logit}^{-1}(y)\big)\cdot(b-a)\cdot \text{logit}^{-1}(y)\cdot\big(1-\text{logit}^{-1}(y)\big)$$

- a = 0, b = 1
- For our model: $\text{logit}^{-1}(y) = \theta$
- Need to take log of the expression above
- lp__ = (target) + log(1 - 0) + log(theta) + log(1 - theta)

Example: automatic transform

- What's the value of lp__ for theta = 0.5?
- What's the value of lp__ for theta = 0.6?

- In R:

```
log(0.5) + log(0.5)
log(0.6) + log(0.4)
```

- `lp__ with theta = 0.5: -1.386294`
- `lp__ with theta = 0.5: -1.427116`

Example: automatic transform

- Run the model!
- ./example-4 sample output file=output-4.csv

```
lp__,accept_stat__,stepsize__,treedepth__,n_leapfrog__,divergent__,energy__,theta
-1.98239,1,1.03392,1,1,0,2.2833,0.164946
-2.25774,0.948028,1.03392,1,1,0,2.3653,0.118668
-1.45006,0.978329,1.03392,1,3,0,2.0142,0.37573
-1.52885,0.972027,1.03392,1,1,0,1.54515,0.317747
```

Thank you!

Stan resources

- http://mc-stan.org
- http://discourse.mc-stan.org
- StanCon Helsinki 2018: Aug 29 - 31

- Daniel Lee

  daniel@generable.com

  @djsyclik

  https://generable.com

Generable