A Roadmap to Developing for Stan

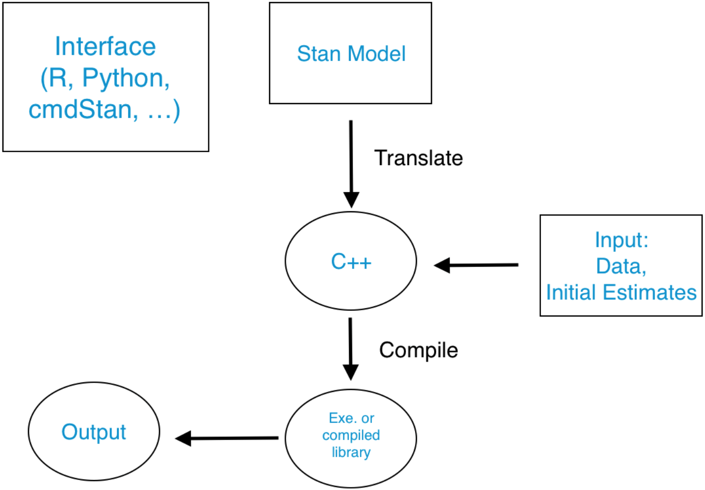
Charles Margossian

Columbia University, Department of Statistics

January 11th 2018

Outline

1. Where does Stan live?
2. Stan's sampler: Hamilton Monte Carlo
3. Efficiently computing gradients with Automatic Differentiation
4. C++ implementation
5. From C++ to the Stan language

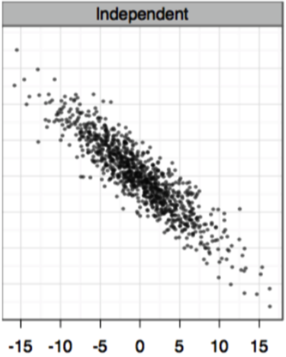

Interface
(R, Python, cmdStan, ...)
Stan Model
Translate
C++
Input:
Data,
Initial Estimates
Compile
Output
Exe. or compiled library

Source Code

- `github.com/stan-dev`
  - math
  - stan
  - rStan, pyStan, ...

1. Where does Stan live?
2. Stan's sampler: Hamilton Monte Carlo
3. Efficiently computing gradients with Automatic Differentiation
4. C++ implementation
5. From C++ to the Stan language

Pathological geometry



Performance of various samplers

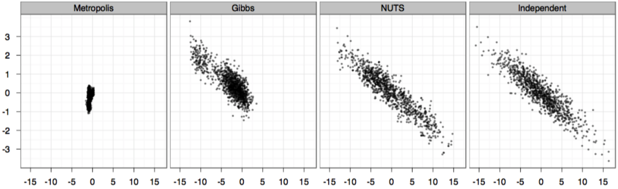


Figure 7: *Samples generated by random-walk Metropolis, Gibbs sampling, and NUTS. The plots compare 1,000 independent draws from a highly correlated 250-dimensional distribution (right) with 1,000,000 samples (thinned to 1,000 samples for display) generated by random-walk Metropolis (left), 1,000,000 samples (thinned to 1,000 samples for display) generated by Gibbs sampling (second from left), and 1,000 samples generated by NUTS (second from right). Only the first two dimensions are shown here.*

For good references, see [1, 2].

So what's the trade-off?

Need to compute:

$$-\nabla \log(\pi(\theta|x))$$

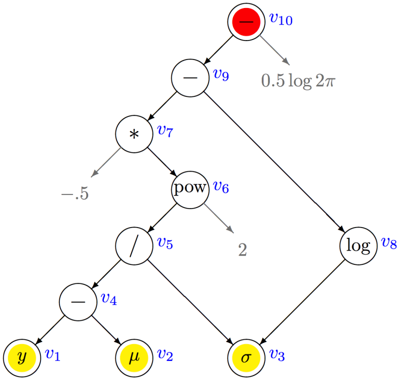
1. Where does Stan live?
2. Stan's sampler: Hamilton Monte Carlo
3. Efficiently computing gradients with Automatic Differentiation
4. C++ implementation
5. From C++ to the Stan language

An example from [3]

$$\log[\mathrm{Normal}(y|\mu, \sigma)] = -\frac{1}{2}\left(\frac{y-\mu}{\sigma}\right)^2 - \log(\sigma) - \frac{1}{2}\log(2\pi)$$

Expression graph

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



Techniques to compute gradients

- Hand-code analytical derivative
- Finite differentiation:

$$\frac{\partial f}{\partial x} = \lim_{h \to 0} \frac{f(x+h) - f(x-h)}{2h}$$

- Symbolic differentiation
- Automatic differentiation

For an excellent review, see [4].

Forward Automatic Differentiation

var	fwd. eval. trace	$\partial/\partial y$		
v_1	$y = 10$	$\dot{v}_1$	$= \dot{y}_1$	$= 1$
v_2	$\mu = 5$	$\dot{v}_2$	$= \dot{\mu}$	$= 0$
v_3	$\sigma = 2$	$\dot{v}_3$	$= \dot{\sigma}$	$= 0$
v_4	$v_1 - v_2 = 5$	$\dot{v}_4$	$= 1$	$= 1$
v_5	$v_4 / v_3 = 2.5$	$\dot{v}_5$	$= 1/v_3 \times \dot{v}_4$	$= 0.5 \times 1$
v_6	$v_5^2 = 6.25$	$\dot{v}_6$	$= 2v_5 \times \dot{v}_5$	$= 2 \times 2.5 \times 0.5$
v_7	$-0.5 \times v_6 = 3.125$	$\dot{v}_7$	$= -0.5 \times \dot{v}_6$	$= -0.5 \times 2.5$
v_8	$\log(\mu) = \log(2)$	$\dot{v}_8$	$= 1/v_3 \times \dot{v}_3$	$= 0$
v_9	$v_7 - v_8 = 3.125 - \log(2)$	$\dot{v}_9$	$= \dot{v}_7 - \dot{v}_8$	$= -1.25 - 0$
v_{10}	$v_9 - 0.5\log(2\pi) = 3.125 - \log(4\pi)$	$\dot{v}_{10}$	$= \dot{v}_7$	$= -1.25$

Reverse automatic differentiation

Define the adjoint of v_i with respect to f:

$$\bar{v}_i = \frac{\partial f}{\partial v_i}$$

Procedure:

- Do a forward evaluation trace.
- Compute derivatives, this time starting at the top of the tree and ending at the roots.

Reverse automatic differentiation II

Reverse adjoint trace		
$\bar{v}_{10}$	$= 1$	$= 1$
$\bar{v}_9$	$= 1 \times \bar{v}_{10}$	$= 1$
$\bar{v}_8$	$= -1 \times \bar{v}_9$	$= -1 \times 1 = -1$
$\bar{v}_7$	$= 1 \times \bar{v}_9$	$= 1 \times 1 = 1$
$\bar{v}_6$	$= -0.5 \times \bar{v}_7$	$= -0.5 \times 1 = -0.5$
$\bar{v}_5$	$= 2 \times v_5 \times \bar{v}_6$	$= 2 \times 2.5 \times -0.5 = -2.5$
$\bar{v}_4$	$= 1/v_3 \bar{v}_5$	$= -1.25$
$\bar{v}_3$	$= -v_4/v_3^2 \times \bar{v}_5 + 1/v_3 \times \bar{v}_8$	$= -5/2^2 \times (-2.5) + 1/2 \times -1 = 2.625$
$\bar{v}_2$	$= -1 \times \bar{v}_4$	$= -1 \times 1.25 = -1.25$
$\bar{v}_1$	$= 1 \times \bar{v}_4$	$= 1 \times 1.25 = 1.25$

Reverse or forward mode?

Consider the map: $f : R^n \rightarrow R^m$.

Which mode should we use:

- when $n >> m$?

Reverse or forward mode?

Consider the map: $f : R^n \rightarrow R^m$.

Which mode should we use:

- when $n << m$?

Reverse or forward mode?

Consider the map: $f : R^n \rightarrow R^m$.

Which mode should we use:

- when $n = m$?

Reverse or forward mode?

Consider the map: $f : R^n \rightarrow R^m$.

Which mode should we use:

- when doing Hamiltonian Monte Carlo sampling?

How could we optimize autodifff?

The `var` class

A `var` object stores:

- the value of the variable: `x.val()`
- its adjoint: `x.adj()`

The `fvar` class is used for forward mode autodiff.

Example: subtract function

Basic C++ implementation:

```
double // return type
subtract(double a, double b) {
  return a - b;
}
```

Example: subtract function

Alternative implementation:

```
inline double // return type
subtract(const double& a, const double& b) {
  return a - b;
}
```

Example: subtract function

Candidate implementation for Stan:

```
inline var // return type
subtract(const var& a, const var& b) {
  return a - b;
}
```

Promoting variables

Consider variables with type `T1` and `T2`.

Define new type:

```
stan::return_type<T1, T2>::type
```

Example: subtract function

Desired implementation:

```
template <typename T1, typename T2>
inline stan::return_type<T1, T2>::type
subtract(const T1& a, const T2& b) {
  return a - b;
}
```

Directories in Stan-Math

- `prim`, `rev`, `fwd`
- `scal`, `arr`, `mat`
- `fun`, `functor`, ...

In which directory should we store `subtract.hpp`?

Expose `subtract` to the relevant header file:

`stan/math/prim/scal.hpp`

Unit Test

Google unit tests:

```
TEST(MathScalar, subtract) {
  using stan::math::subtract;
  double a = 1, b = 2;
  EXPECT_EQ(-1, subtract(a, b));
}
```

Unit Test

A more complete test also checks gradient evaluation:

```
TEST(MathScalar, subtract_grad) {
  using stan::math::var;
  using stan::math::subtract;
  var a = 1, b = 2;
  var f = subtract(a, b);
  EXPECT_EQ(-1, f.val());

  std::vector<double> g;
  std::vector<var> x = createAVEC(a, b);
  f.grad(x, g);
  EXPECT_EQ(1, g[0]);
  EXPECT_EQ(-1, g[1]);
}
```

How would we test more sophisticated gradients?

To run the unit test from the command line:

```
./runTests.py test/unit/math/rev/scal/fun/subtract_test.cpp
```

1. Where does Stan live?
2. Stan's sampler: Hamilton Monte Carlo
3. Efficiently computing gradients with Automatic Differentiation
4. C++ implementation
5. From C++ to the Stan language

Expose the signature(s) of the function

Go to the `stan` repo.

In `src/stan/lang/function_signatures.h`, add the following line:

```
add("subtract", expr_type(double_type()),
    expr_type(double_type()),
    expr_type(double_type()));
```

Unit test in stan

In src/test/unit/lang/parser/math_functions_test.cpp:

```
TEST(lang_parser, subtract_math_function_signatures) {
  test_parsable("function-signatures/math/functions/subtract");
}
```

Unit test in stan

In `src/test/test-models/good/function-signatures/math/functions`:

```
data {
 real a;
 real b;
}

transformed data {
 real f = subtract(a, b);
}

parameters {
 real a_p;
 real b_p;
}

. . .
```

Unit test in stan

```
. . .

transformed parameters {
 real f_p = subtract(a, b);
 f_p = subtract(a_p, b);
 f_p = subtract(a, b_p);
 f_p = subtract(a_p, b_p);
}

model {
 a_p ~ normal(0, 1);
}
```

What didn't we cover?

In `math`:

- Error messages:
  - Invalid arguments
  - Invalid metropolis proposal
- Incorporating analytical derivatives.
- Expected coding practices.

What didn't we cover?

In `stan`:

- Exposing higher-order functions

Q & A

References I

[1] Michael Betancourt.
A conceptual introduction to hamiltonian monte carlo.
arXiv:1701.02434v1, January 2017.

[2] Matthew D. Hoffman and Andrew Gelman.
The no-u-turn sampler: Adaptively setting path lengths in hamiltonian monte carlo.
Journal of Machine Learning Research, pages 1593–1623, April 2014.

[3] Bob Carpenter, Matthew D. Hoffman, Marcus A. Brubaker, Daniel Lee, Peter Li, and Michael J. Betancourt.
The stan math library: Reverse-mode automatic differentiation in c++.
arXiv 1509.07164., 2015.

[4] Atilim G. Baydin, Barak A. Pearlmutter, Alexey A. Radul, and Sisking Jeffrey M.
Automatic differentiation in machine learning: a survey.
arXiv:1502.05767v2, April 2015.