Probabilistic Programming

and Bayesian Inference with Stan

Bob Carpenter

Center for Computational Mathematics

Flatiron Institute

October 2020

`https://mc-stan.org`

Stan

Uncertainty

- **permeates science** and decision making
- in the form of
  - **sampling** uncertainty,
  - **measurement** uncertainty, and
  - **modeling** uncertainty.

- The **alternative to good statistics** is
  - not no statistics,
  - but **bad statistics**. (Bill James)

Probability & Statistics

- Probability theory uses math to **quantify uncertainty**.
- **Bayesian statistics** applies probability theory to
  - data analysis,
  - prediction & forecasting,
  - model evaluation, and
  - decision theory.
- The **computational bottlenecks** are
  - model expression and
  - posterior inference.

Probability isn't . . .

An intellect which at a certain moment would know all forces that set nature in motion, and all positions of all items of which nature is composed, if this intellect were also vast enough to submit these data to analysis, it would embrace in a single formula the movements of the greatest bodies of the universe and those of the tiniest atom; **for such an intellect nothing would be uncertain** and the future just like the past would be present before its eyes.

– **Pierre-Simon Laplace**. 1814. *A Philosophical Essay on Probabilities*. Translated by F. W. Truscott and F. L. Emory, 1951.

Probability is . . .

Every event is in itself certain, not probable; if we knew all, we should either know positively that it will happen, or positively that it will not. But its **probability** to us means the **degree of expectation of its occurrence, which we are warranted in entertaining by our present evidence**.

– **John Stuart Mill**. 1882. *A System of Logic: Raciocinative and Inductive*. Eighth edition. III:18.

The chance of rain in Edinburgh?

- What's the chance of rain in Edinburgh, Scotland?
  - if it could be any day of the year: 52%
  - if I know the month is October: 55%
  - if I also know it's 24h from now: 55%
  - if I also have today's weather report: 80%
  - if it's now and I see rain: 100%
- The world didn't change.
- But, our knowledge changed.

Bayesian Inference

Notation

- Variables
  - y : **data** (observed)
  - θ : **parameters** (unknown)

- Probability functions
  - $p(y, \theta)$: **joint** density
  - $p(y \mid \theta)$: **sampling** density (**likelihood** as fun of θ)
  - $p(\theta)$: **prior** (parameter marginal)
  - $p(\theta \mid y)$: **posterior**
  - $p(y)$: **evidence** (data marginal)

Bayes's rule

$$
\begin{aligned}
p(\theta \mid y) &= \frac{p(y,\theta)}{p(y)} \\
&= \frac{p(y \mid \theta) \cdot p(\theta)}{p(y)} \\
&= \frac{p(y \mid \theta) \cdot p(\theta)}{\int_{\Theta} p(y,\theta)\, \mathrm{d}\theta} \\
&= \frac{p(y \mid \theta) \cdot p(\theta)}{\int_{\Theta} p(y \mid \theta) \cdot p(\theta)\, \mathrm{d}\theta} \\
&\propto p(y \mid \theta) \cdot p(\theta).
\end{aligned}
$$

- **Posterior** proportional to **likelihood times prior**

Estimates, events, and predictions

- **Parameter estimate**

$$\hat{\theta} \ = \ \mathbb{E}[\theta \mid y] \ = \ \int_{\Theta} \theta \cdot p(\theta \mid y) \, \mathrm{d}\theta$$

- **Event probability** $(A \subseteq \Theta, \quad \text{e.g., } A = \{\theta : \theta > 0\})$

$$\Pr[A \mid y] \ = \ \mathbb{E}[\mathrm{I}_A(\theta) \mid y] \ = \ \int_{\Theta} \mathrm{I}_A(\theta) \cdot p(\theta \mid y) \, \mathrm{d}\theta$$

- **Predictive inference**

$$p(\tilde{y} \mid y) \ = \ \mathbb{E}[p(\tilde{y} \mid \theta) \mid y] \ = \ \int_{\Theta} p(\tilde{y} \mid \theta) \cdot p(\theta \mid y) \, \mathrm{d}\theta$$

(Markov chain) Monte Carlo

- Given **sample** $\theta^{(1)}, \ldots, \theta^{(M)} \sim p(\theta \mid y)$.
- General **plug-in expectation** calculations,

$$\begin{aligned}
\mathbb{E}[f(\theta) \mid y] &= \int_{\Theta} f(\theta) \cdot p(\theta \mid y) \, \mathrm{d}\theta \\
&= \lim_{M \to \infty} \frac{1}{M} \sum_{m=1}^{M} f\left(\theta^{(m)}\right) \\
&\approx \frac{1}{M} \sum_{m=1}^{M} f\left(\theta^{(m)}\right).
\end{aligned}$$

- (MCMC) central limit theorem **approx. error** $\propto \dfrac{1}{\sqrt{M_{(\text{eff})}}}$

Simulated Binary Data

- True .300 batting average (BA); simulate 10 at bats (AB)
  - **10 at bats** (.000 to .600 BA)

    1 1 2 6 2 0 4 4 4 2
  - **100 at bats** (0.220 to 0.330 BA)

    22 34 30 26 25 31 31 28 27 33
  - **1000 at bats** (.294 to .320 BA)

    296 307 302 298 290 297 303 294 280 320
  - **10,000 at bats** (.295 to .305 BA)

    3022 2953 2987 2946 2994 2989 3052 3024 3038 2938

Decision Theory

Expected utility and risk

- Agent has choice of **actions** $a \in A$.
  - discrete: do I go to college?
  - continuous: how much do I invest for retirement?
- **Utility**: $\mathrm{util}(a, \theta)$ based on state of world θ
  - risk aversion, etc. rolled into utility
- **Expected Utility** conditioned on y:

$$\mathbb{E}[\mathrm{util}(a, \theta) \mid y] \ = \ \int_{\Theta} \mathrm{util}(a, \theta) \cdot p(\theta \mid y) \, \mathrm{d}\theta.$$

- point estimates aren't plug & play due to **non-linearity**

Optimal Decisions

- Given observed data y and model $p(y, \theta)$, choose action $a^* \in A$ that **maximizes expected utility**

$$a^* = \text{arg max}_{\theta}\ \mathbb{E}[\text{util}(a, \theta) \mid y].$$

- **Compute** expectation via MCMC sample

$$\theta^{(1)}, \ldots, \theta^{(m)} \sim p(\theta \mid y),$$

as

$$\mathbb{E}[\text{util}(a, \theta) \mid y] \approx \frac{1}{M} \sum_{m=1}^{M} \text{util}\Big(a, \theta^{(m)}\Big)$$

- can be **automatically differentiated**.

Online A/B/... testing

- discrete choice of action $k \in 1:K$
- probabilistic outcome of action k as $p(y \mid \theta_k)$
- assume utility $\mathtt{util}(\theta, a)$
- observe history of actions $a_1, \ldots, a_t \in 1:K$ and paired outcomes $y_1, \ldots, y_t \in \mathbb{R}$.
- explore/exploit: next action a_{t+1} chosen w. probability proportional to k being best (Thompson sampling)

e.g., COVID-19 and antibody testing risk reduction.

Bayesian Workflow

1. design experiment & collect data (its own workflow)

2. for gradually **increasing model complexity**
    - **test model** and sofware on simulated data (**does it work?**),
    - perform prior predictive checks (**is prior sensible?**)
    - **fit model** to real data,
    - perform posterior predictive checks (**does it fit data?**), and
    - perform held-out checks/cross-validation (**does it predict?**).

Calibration

Calibration

- Empirical (frequentist) evaluation of **probabilistic forecasts**
- Of the days on which a 75% of rain is forecast, roughly 75% of them should be rainy.
  - for N **calibrated predictions** of probability θ, $Y \sim \mathrm{binomial}(N, \theta)$ will obtain
- Calibrated predictions for **rain in Edinburgh**:
  - 52% is calibrated if evaluated over random days of the year.
  - Prediction based on average for day of month is calibrated (i.e., 65% for December)
  - Is `weather.com`'s forecast calibrated? Empirical question.

Sharpness

- Sharp probabilistic predictions have **low entropy**.
- **Narrowly concentrated** predictions are sharper
  - $\Pr[\alpha \in (.58, .59)] = .9$ sharper than $\Pr[\alpha \in (.5, .6)] = .9$
- Probabilistic predictions **nearer 0 or 1** are sharper
  - $\Pr[z_i = 1] = 0.95$ sharper than $\Pr[z_i = 1] = 0.6$

Inference evaluation criterion

- For probabilistic forecasts:

  Maximize sharpness subject to calibration.

- **Goal**: consistent, useful inference from **finite data**.
  - cf. minimize variance subject to zero bias for estimators
- Bayesian models are calibrated if they are **well-specified**
  - i.e., they capture the true data-generating process.
  - But, models are always approximate.
  - So, need to test fit to real data and predictions.

Example 1

Birth Ratio

Birth ratio (Laplace, 1781)

- **Live births** in Paris, 1745–1770
  - $y = 251{,}527$ male, $N = 493{,}472$ total
- **Sampling**: $p(y \mid N, \theta) = \mathrm{binomial}(y \mid N, \theta)$.
- **Prior**: $p(\theta) = \mathrm{uniform}(\theta \mid 0, 1)$
- **Posterior**: $p(\theta \mid y) = \mathrm{beta}(\theta \mid 1 + y,\ 1 + N - y)$.

- $\Pr[\theta \in (0.508, 0.512)] = 0.99$ (estimated **male birth rate**)
- $\Pr[\theta > 0.5] \approx 1 - 10^{-42}$ ("morally certain" **more boys**)

Coding in Stan

```
transformed data {
  int y = 251527;  int N = 493472;
}
parameters {
  real<lower=0, upper=1> theta;
}
model {
  y ~ binomial(N, theta);
}
generated quantities {
  int<lower=0, upper=1> theta_gt_half = (theta > 0.5);
  int<lower = 0> y_sim = binomial_rng(100, theta);
}
```

Executing

```
> fit <- stan("laplace.stan")
> print(fit, probs=c(0.005, 0.995), digits=3)

                    mean se_mean   0.5%   99.5%
theta              0.510   0.000  0.508   0.512
theta_gt_half      1.000     NaN  1.000   1.000
y_sim             50.902   0.081 38.000  64.000
```

- **estimate** $\hat{\theta}$ is posterior sample **mean** for θ;
- 99% **interval** for θ estimated by sample **quantiles**
- $\mathrm{Pr}[\theta > 0.5]$ estimated by sample **mean of indicator**
- y_sim estimated **forecast** for next 100 births

Example 2

Population Dynamics

Population Dynamics (Volterra 1926)

- populations at t of prey $u(t)$ & predator $v(t)$
- Volterra's mechanistic model

$$\frac{\mathrm{d}}{\mathrm{d}t}u = (\alpha - \beta \cdot v) \cdot u \qquad \frac{\mathrm{d}}{\mathrm{d}t}v = (-\gamma + \delta \cdot u) \cdot v$$

  – α: prey growth rate; β: predation shrinkage

  – γ: predator shrinkage; δ: predation growth

Analytic solution (Volterra 1926)

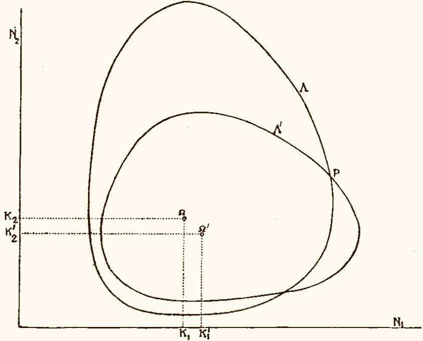
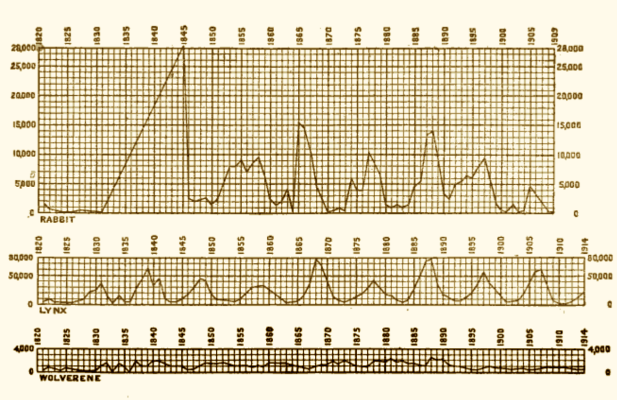


Fig. 1.

Measurement/model error

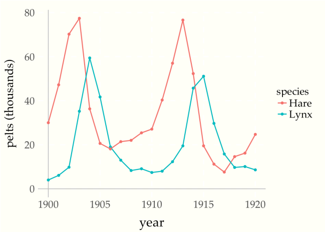
- u, v are observed
- $\hat{u}, \hat{v}$ predicted by mechanistic model
- Independent error proportional to population size
  - $u_t \sim \mathrm{lognormal}(\hat{u}_t, \sigma_1)$
  - $v_t \sim \mathrm{lognormal}(\hat{v}_t, \sigma_2)$
- Weakly informative priors determine scale
  - $\alpha, \gamma \sim \mathrm{normal}(1, 0.5); \quad \beta, \delta \sim \mathrm{normal}(0.05, 0.05)$
  - $\sigma_1, \sigma_2 \sim \mathrm{lognormal}(-1, 1)$

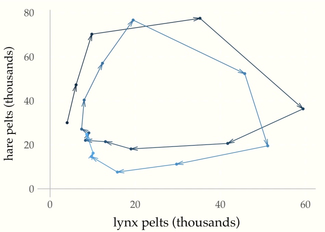
Hudson’s Bay Co. pelts (Hewitt 1921)



PERIODIC FLUCTUATIONS OF RABBIT, LYNX, AND WOLVERENE IN CANADA

Plotted pelt data



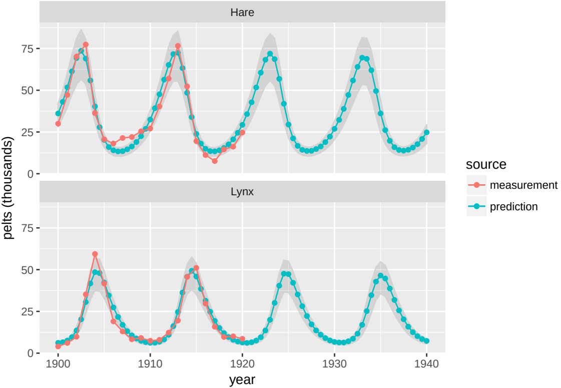


- left) pelts vs. time (line per species)
  right) hare vs. lynx pelts (over time)
- pelt data proxy proportional to population (could model)
- as is, model predicts pelts, not population

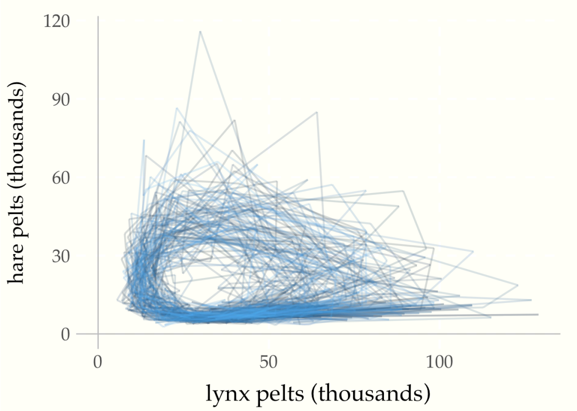
Stan dynamics & error

```
real[] dz_dt(real t, real[] uv, real[] theta,
             real[] theta, real[] x_r, int[] x_i) {
  return { (theta[1] - theta[2] * uv[2]) * uv[1],
           (-theta[3] + theta[4] * uv[1]) * uv[2] };
}
...
real z[T, 2]
  = integrate_ode(dz_dt, z0, t0, sol_ts[1:T], theta,
                  rep_array(0.0, 0), rep_array(0, 0));
...
y[1:T, 1] ~ lognormal(z[1:T, 1], sigma_u);
y[1:T, 2] ~ lognormal(z[1:T, 2], sigma_v);
```

Measurements & predictions



100 predictive draws



Summary

- mechanistic forward model of dyamics (Lotka 1926)
  - $\hat{y} = f(\theta)$
- error model $y_{t,k} \sim \mathrm{lognormal}(\hat{y}_{t,k}, \sigma_k)$
- added weakly informative priors $p(\theta)$ (fixing scale)
- turn the Bayesian crank to sample from posterior
  - $\theta^{(1)}, \ldots, \theta^{(M)} \sim p(\theta \mid y)$.
- This methodology **works for many scientific models**.

Example 3

Linear Models

Logistic regression

```
data {
  int<lower = 1> K;
  int<lower = 0> N;
  matrix[N, K] x;                   // predictors
  int<lower=0, upper=1> y[N];       // observations
}
parameters {
  vector[K] beta;                   // regression coeffs
}
model {
  beta ~ normal(0, 2);             // prior
  y ~ bernoulli_logit(x * beta);   // likelihood
}
```

Linear regression with prediction

```
data {
  int<lower=0> K;
  int<lower=0> N;             int<lower=0> N_tilde;
  matrix[N, K] x;             matrix[N_tilde, K] x_tilde;
  vector[N] y;
}
parameters {
  vector[K] beta;             real<lower=0> sigma;
}
model {
  y ~ normal(x * beta, sigma);
}
generated quantities {
  vector[N_tilde] y_tilde
    = normal_rng(x_tilde * beta, sigma);
}
```

Time series autoregressive: AR(1)

```
data {
  int<lower=0> N;   vector[N] y;
}
parameters {
  real alpha;  real beta;  real sigma;
}
model {
  y[2:n] ~ normal(alpha + beta * y[1:(n-1)], sigma);
}
```

Priors with covariance

- G groups w. varying slope and intercept; `gg[n]` indicates group

- LKJ prior concentrates around unit correlation ($\propto \det(\Omega)^{\eta-1}$)
- Cholesky-factor parameterization for efficiency ($\Omega = \mathrm{L}_\Omega \cdot \mathrm{L}_\Omega^\top$)

```
parameters {
  vector[2] beta[G];
  cholesky_factor_corr[2] L_Omega;
  vector<lower = 0>[2] sigma;

model {
  matrix[2, 2] L_Sigma = diag_pre_multiply(sigma, L_Omega
  sigma ~ normal(0, 2);
  L_Omega ~ lkj_cholesky(4);
  beta ~ multi_normal_cholesky(zeros(2), L_Sigma);

  y ~ bernoulli_logit(... + x .* beta[gg]);
```

Gaussian process

```
data {
  int<lower=1> N;  vector[N] x; vector[N] y;
} parameters {
  real<lower=0> eta_sq, inv_rho_sq, sigma_sq;
} transformed parameters {
  real<lower=0> rho_sq; rho_sq = inv(inv_rho_sq);
} model {
  matrix[N,N] Sigma;
  for (i in 1:(N-1)) {
    for (j in (i+1):N) {
      Sigma[i,j] = eta_sq * exp(-rho_sq * square(x[i] - x
      Sigma[j,i] = Sigma[i,j];
  }}
  for (k in 1:N) Sigma[k,k] = eta_sq + sigma_sq;
  eta_sq, inv_rho_sq, sigma_sq ~ cauchy(0,5);
  y ~ multi_normal(rep_vector(0,N), Sigma);
}
```

Stan

What is Stan?

- a domain-specific **probabilistic programming language**
- Stan **program** defines a **differentiable** probability model
  - declares data and (constrained) parameter variables
  - defines log posterior (or penalized likelihood)
  - defines predictive quantities
- Stan **inference** fits model & makes predictions
  - MCMC for full Bayesian inference
  - variational and Laplace for approximate Bayes
- Carpenter, B., Gelman, A., Hoffman, M. D., Lee, D., Goodrich, B., Betancourt, M., Marcus Brubaker, Jiqiang Guo, Peter Li, Riddell, A. (2017). Stan: A probabilistic programming language. *J. Stat. Soft.* 76(1).

Availability & Usage

- *Platforms:* Linux, Mac OS X, Windows
- *Interfaces:* R, Python, Julia, MATLAB, Mathematica
- *Developers (academia & industry):* 40+ (15+ FTEs)
- *Users:* tens or hundreds of thousands
- *Companies using:* hundreds or thousands
- *Downloads:* millions
- *User's Group:* 3000+ registered; 6000+ non-bot views/day
- *Books using:* 10+
- *Courses using:* 100+
- *Case studies about:* 100+
- *Articles using:* 3000+
- *Conferences:* 4 (800+ attendance)

Some published applications

- **Physical sciences**: astrophysics, statistical mechanics, particle physics, (organic) chemistry, geology, oceanography, climatology, biogeochemistry, materials science, ...
- **Biological sciences**: molecular biology, clinical drug trials, entomology, pharmacology, toxicology, opthalmology, neurology, genomics, agriculture, botany, fisheries, epidemiology, population ecology, neurology, psychiatry, ...
- **Social sciences**: econometrics (macro and micro), population dynamics, cognitive science, psycholinguistics, social networks, political science, survey sampling, anthropology, sociology, social work, ...
- **Other**: education, public health, A/B testing, government, finance, machine learning, logistics, electrical engineering, transportation, actuarial science, sports, advertising, marketing, ...

Industries using Stan

- **marketing attribution**: Google, Domino's Pizza, Legendary Ent.
- **demand forecasting**: Facebook, Salesforce
- **financial modeling**: Two Sigma, Point72
- **pharmacology & CTs**: Novartis, Pfizer, Astra Zeneca
- **(e-)sports analytics**: Tampa Bay Rays, NBA, Sony Playstation
- **survey sampling**: YouGov, Catalist
- **agronomy**: Climate Corp., CiBO Analytics
- **real estate pricing models**: Reaktor
- **industrial process control**: Fero Labs

Why is Stan so Popular?

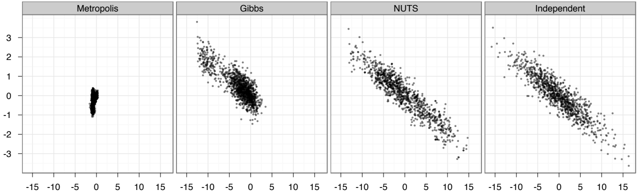
- **Community**: large, friendly, helpful, and sharing
- **Documentation**: novice to expert; breadth of fields
- **Robustness**: industrial-strength code; user diagnostics
- **Flexibility**: highly expressive language; large math lib
- **Portability**: popular OS, language, and cloud support
- **Extensibility**: developer friendly; derived packages
- **Speed**: $2 - \infty$ orders of magnitude faster
- **Scalability**: $2+$ orders of magnitude more scalable
- **Openness**: permissive code and doc licensing

What Stan Does

No-U-turn sampler

- **Hamiltonian Monte Carlo** (HMC)
  - **Potential Energy**: negative log posterior $-\log p(\theta \mid y)$
  - **Kinetic Energy**: random standard normal each iteration
  - **geometric ergodicity** for wider range of models & data
- Adapt leapfrog algorithm **during warmup**
  - step size adapted to target acceptance rate
  - Euclidean metric estimated w. sample covariance
- Adapt leapfrog algorithm **during sampling**
  - simulate forward and backward in time until U-turn
  - multinomial sample trajectory propto energy; bias furthest
- developed concurrently w. Stan (Hoffman & Gelman 2011)

NUTS vs. Gibbs and Metropolis



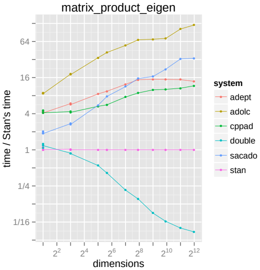
- Two dimensions from highly correlated 250-dim normal
- **1,000,000 draws** from Metropolis and Gibbs (thinned to 1000)
- **1000 draws** from NUTS; 1000 independent draws

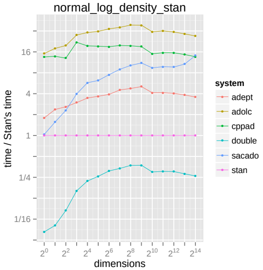
Reverse-mode autodiff (adjoint)

- Build up expression graph with values x and adjoints $\overline{x}$
- Differentiating $f : \mathbb{R}^N \rightarrow \mathbb{R}$ is additional $\mathcal{O}(1)$
- Set final result y's adjoint $\overline{y} = 1$ and proceed in reverse
- Example: scalar $c = \log a$
  - $\overline{a} \mathrel{+}= \overline{c} \cdot \frac{1}{a}$
- Example: scalar $c = a \cdot b$
  - $\overline{a} \mathrel{+}= \overline{c} \cdot b; \quad \overline{b} \mathrel{+}= \overline{c} \cdot a$
- Example: matrix $C = A^{-1}$
  - $\overline{A} \mathrel{+}= -C^\top \cdot \overline{C} \cdot C^\top$

Stan's Autodiff vs. Alternatives

- Stan is **fastest** (and uses least memory)
  - among open-source C++ alternatives





- Carpenter, B. et al. (2015). The Stan math library: Reverse-mode automatic differentiation in C++. *arXiv:*1509.07164

Forward-mode autodiff (tangent)

- Build tangents forward with values x and tangents $\dot{x}$
- Differentiating $f : \mathbb{R} \rightarrow \mathbb{R}^M$ is additional $\mathcal{O}(1)$
- To differentiate w.r.t. x, set $\dot{x} = 1$ and work forward
- Example: scalar $c = \log a$
  - $\dot{c} = \dot{a} \cdot \frac{1}{a}$
- Example: scalar $c = a \cdot b$
  - $\dot{c} = \dot{a} \cdot b + \dot{b} \cdot a$
- Example: matrix $C = A^{-1}$
  - $\dot{C} = -C \cdot \dot{A} \cdot C$

Higher-order autodiff

- Open up new algorithms exploiting curvature
  - Riemannian HMC, nested Laplace approximations (ala INLA), nested Jacobians for ODE solver, Newton solvers for optimizers and nested algebraic equations
- Considering log densities $f : \mathbb{R}^N \to \mathbb{R}$
- Second-order derivatives
  - nest reverse in forward adds $\mathcal{O}(N)$
  - nest forward in forward $\mathcal{O}(N^2)$
- Third-order derivatives
  - nest reverse in forward in forward $\mathcal{O}(N^2)$
  - nest forward in forward in forward $\mathcal{O}(N^3)$

Stan's shortcomings

- Failures
  - extreme varying posterior curvature (stiff—can't adapt)
  - floating point (e.g., log cdfs and ccdfs, gradients)
  - scalability of data and parameters
- Missing features
  - black-box log densities (cf. emcee in Python)
  - checkpointd autodiff for large Gaussian processes
  - filtering (online model updating)
  - stochastic algebraic & differential eqs, PDEs
  - structured data types, sparse data types & closures

Discrete parameters?

- **Stan's focus**: tractably **marginalized**
  - e.g., mixtures, HMMs, state-space models
  - **efficient** in theory due to Rao-Blackwell; in practice by eliminating combinatorics; much better tail behavior
  - marginalizations available wherever you find EM & MML
- **Out of scope**: combinatorially **intractable**
  - e.g., selection, clustering, random trees, neural nets
  - optimization is NP-hard or worse
  - thus **nobody knows how to get right** answer in general
  - some special cases can be fit
- **In between**: missing count data

Canned Answers

What is AI?

- AI is about pushing frontier of “thinking” computers
  - playing Go, driving a car, translating natural language, recognizing images, . . .
- AI is not about a technique
  - according to Google, regression is AI!
- ML is more difficult to characterize

ML and Stats (1)

- Traditional stats is about data analysis
- ML is about prediction
  - this is why ML is eating stats' lunch
- Stats is about calibrated uncertainty quantification
  - for estimators, traditional hypothesis testing
  - for predictions, Bayesian posteriors
- Bayesian stats is about prediction *and* data analysis

ML and Stats (2)

- Models
  - Stats focuses on interpretable, tractable parametric models
  - ML focuses on black-box, intractable, non-parametric models
- Inference
  - ML focuses on prediction based on point estimates
  - traditional stats focuses on estimation and testing
  - Bayesian stats on posterior uncertainty quantification

ML ande Stats (3)

- Scale
  - ML only works well with big data
  - Stats tries to squeeze info out of small data
  - Stats builds more model with more data (spatio-temporal, sub-population, etc.)

TensorFlow or PyTorch or JAX?

- less expressive than Stan’s math library
- faster than Stan on GPU/TPU, slower on CPU
- Focus of all of these libraries is
  - approximate algorithms and
  - neural networks.
- Biggest difference is static vs. dynamic autodiff
  - Pytorch/JAX are dynamic like Stan
  - TensorFlow optimizes static autodiff
  - Theano (PyMC3) symbolic

TensorFlow Probability?

- embedded in Python
- adds stats functions to TensorFlow
- catching up to Stan math lib functionality for general models
- more expressive for neural networks
- adds Bayesian inference algorithms to TensorFlow
- not focused on user language (Edward2 & PyMC4 moribund)

PyMC3

- embedded in Python
  - variables are composable Python objects
  - NumPy punned inside PyMC3 namespace
- Theano analytic diff limits control flow
- more flexible sampling algorithm combination (e.g., discrete)