Stanによるハミルトニアンモンテカルロ法を用いたサンプリングについて

10月22日

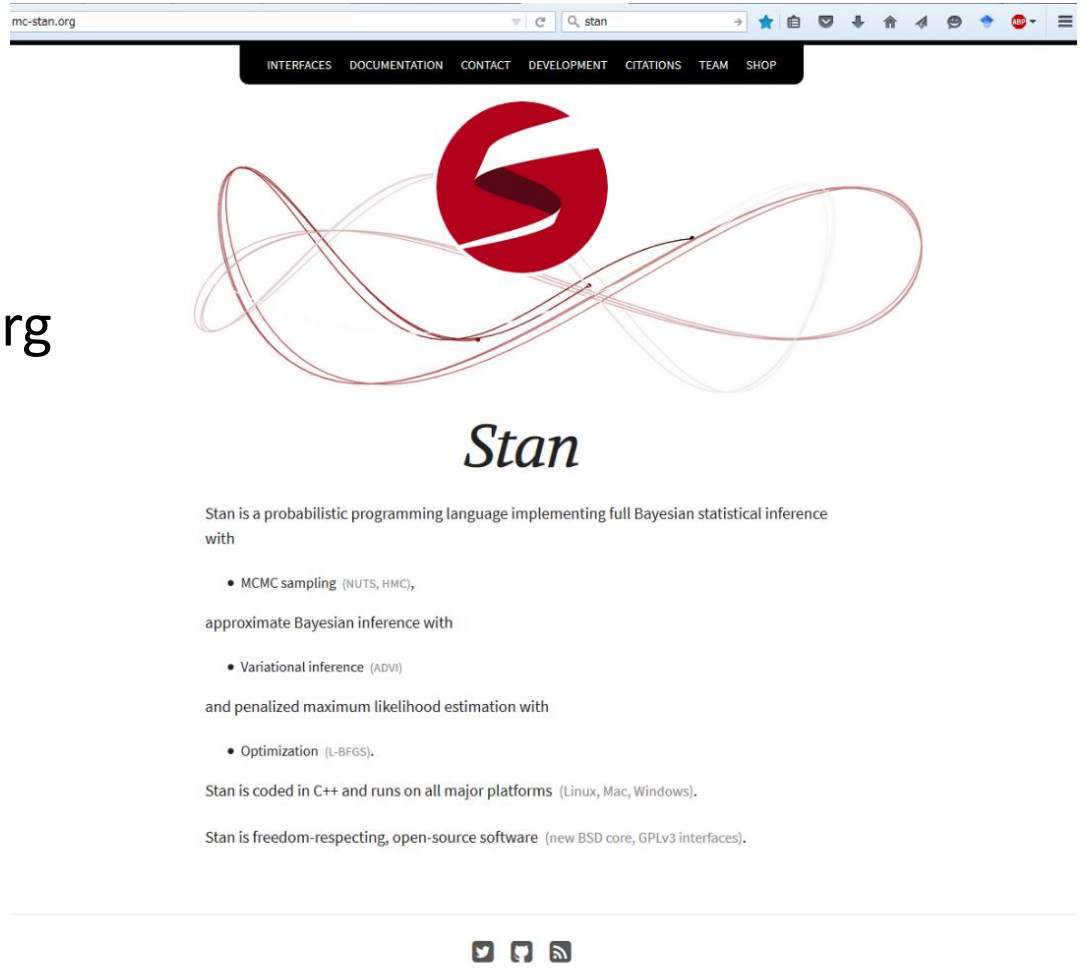
中村　文士

目次

1．STANについて

ハミルトニアンモンテカルロ法に基づいた事後分布からのサンプリングなどができる

STANのHP: mc-stan.org



由来

Stanislaw Ulam（モンテカルロ法の考案した人）の頭文字から

使い方

Stanのプログラミング言語でデータやモデルを記述することでサンプリング

特徴

・StanのコードをC++に変換してC++上でコンパイル、実行をしている
・自動で微分が行われる（ハミルトニアンモンテカルロ法で微分が必要）
・いくつかのプログラミング言語からStanのコードを呼びせる
・オープンソースソフト（GitHub）

事後分布からサンプリングしてやりたいことの例

学習データ $x^n = (x_1, \ldots, x_n)$　学習モデル $p(x|w)$　パラメータの事前分布 $\varphi(w)$

パラメータの事後分布 $p(w|x^n) \propto \prod_{i=1}^{n} p(x_i|w)\varphi(w)$　　$w_j \sim p(w|x^n)$

予測分布

$$p(x|x^n) = \int p(x|w)p(w|x^n)dw \approx \frac{1}{L}\sum_{j=1}^{L} p\left(x|w^{(j)}\right)$$

クロスバリデーション

$$CV = \frac{1}{n}\sum_{i=1}^{n} \log E_w\left[\frac{1}{p(x_i|w)}\right] \approx \frac{1}{n}\sum_{i=1}^{n} \log\left(\frac{1}{L}\sum_{j=1}^{L}\frac{1}{p(x_i|w^{(j)})}\right)$$

予測損失

$$G = -\int q(x)\log E_w[p(x|w)] \approx -\frac{1}{M}\sum_{m=1}^{M} q(x_m)\log\left(\frac{1}{L}\sum_{j=1}^{L} p\left(x|w^{(j)}\right)\right)$$

など

Stanのコードが使えるプログラミング言語

1.コマンドライン（CmdStan）

2.R（RStan）

3.Python（PyStan）

4.Matlab（MatlabStan）

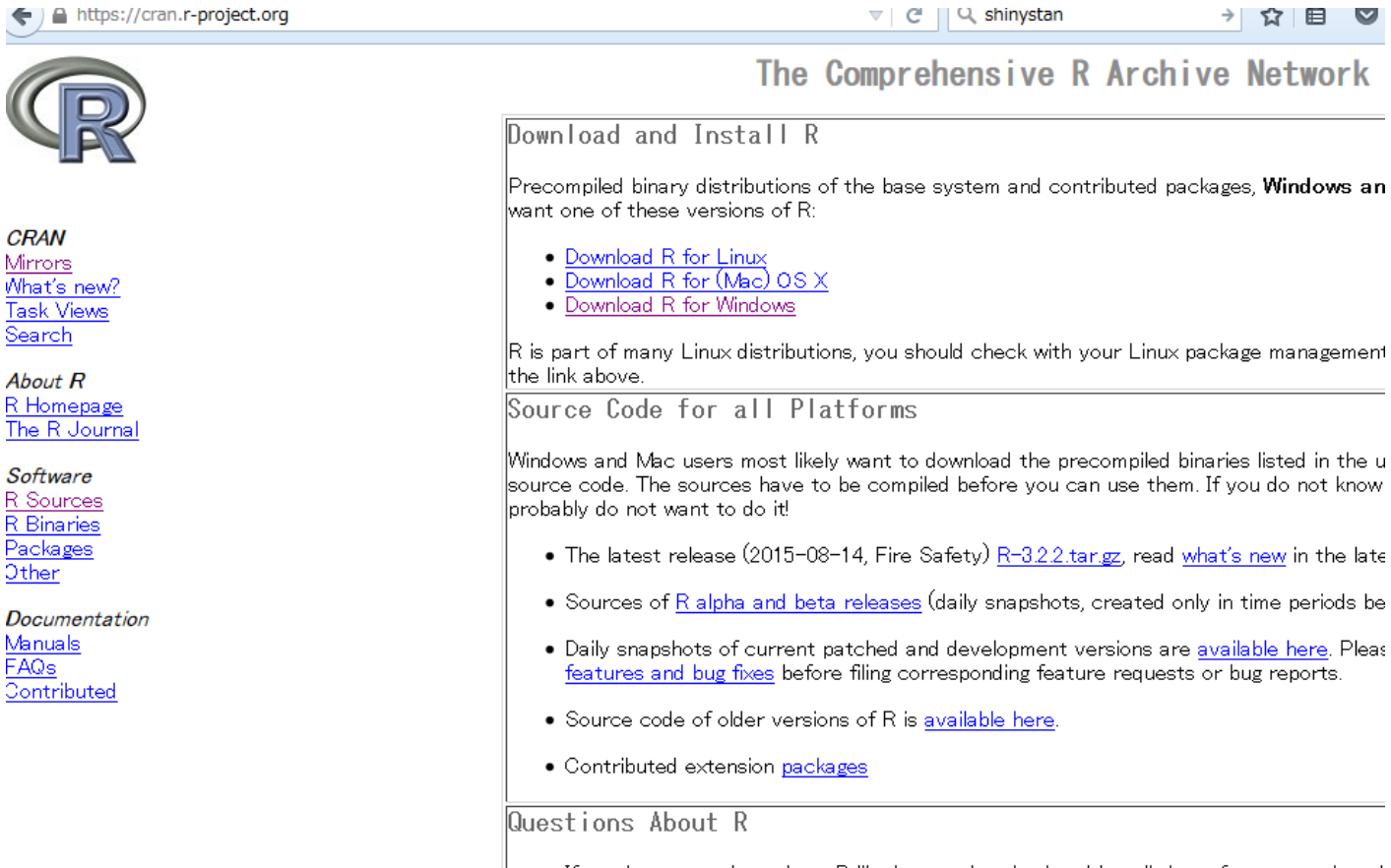
5.Julia（Stan.jl）

6.Stata（StataStan）

2. RStanのインストール

1. Rをインストール

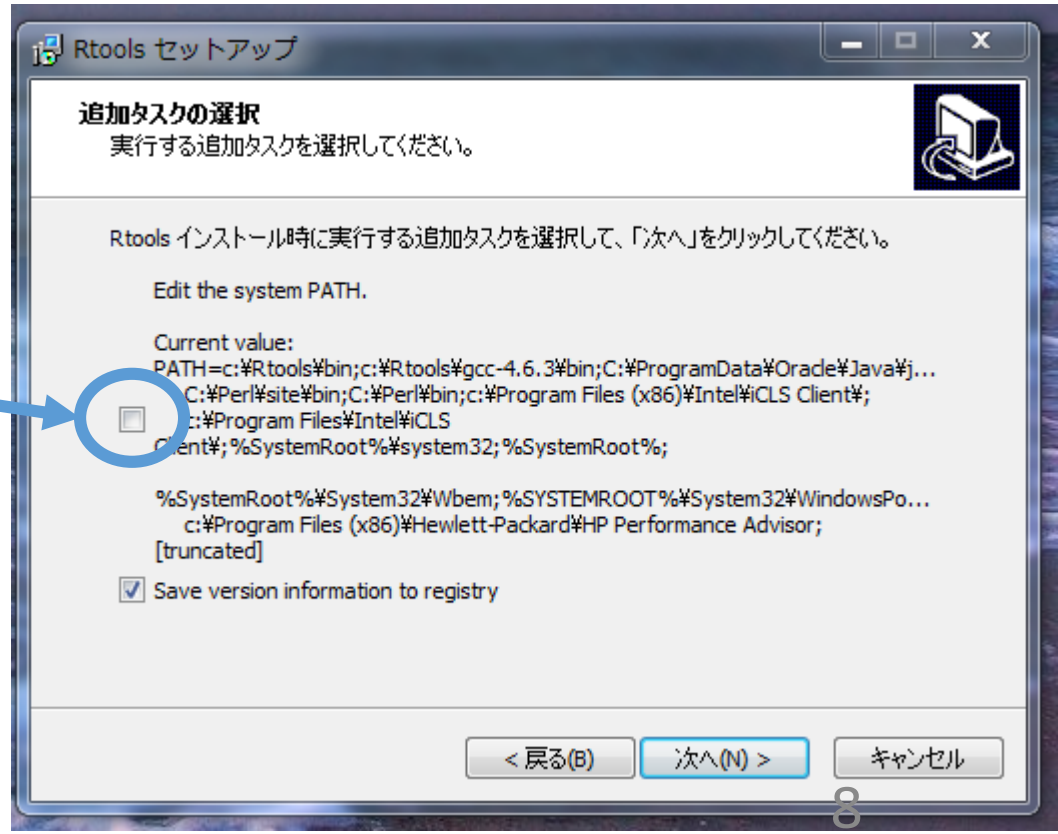
CRAN（https://cran.r-project.org）
やそのミラーサイト（http://cran.ism.ac.jpなど）から対応するOSのインストーラをダウンロードする



2．RToolsをインストール

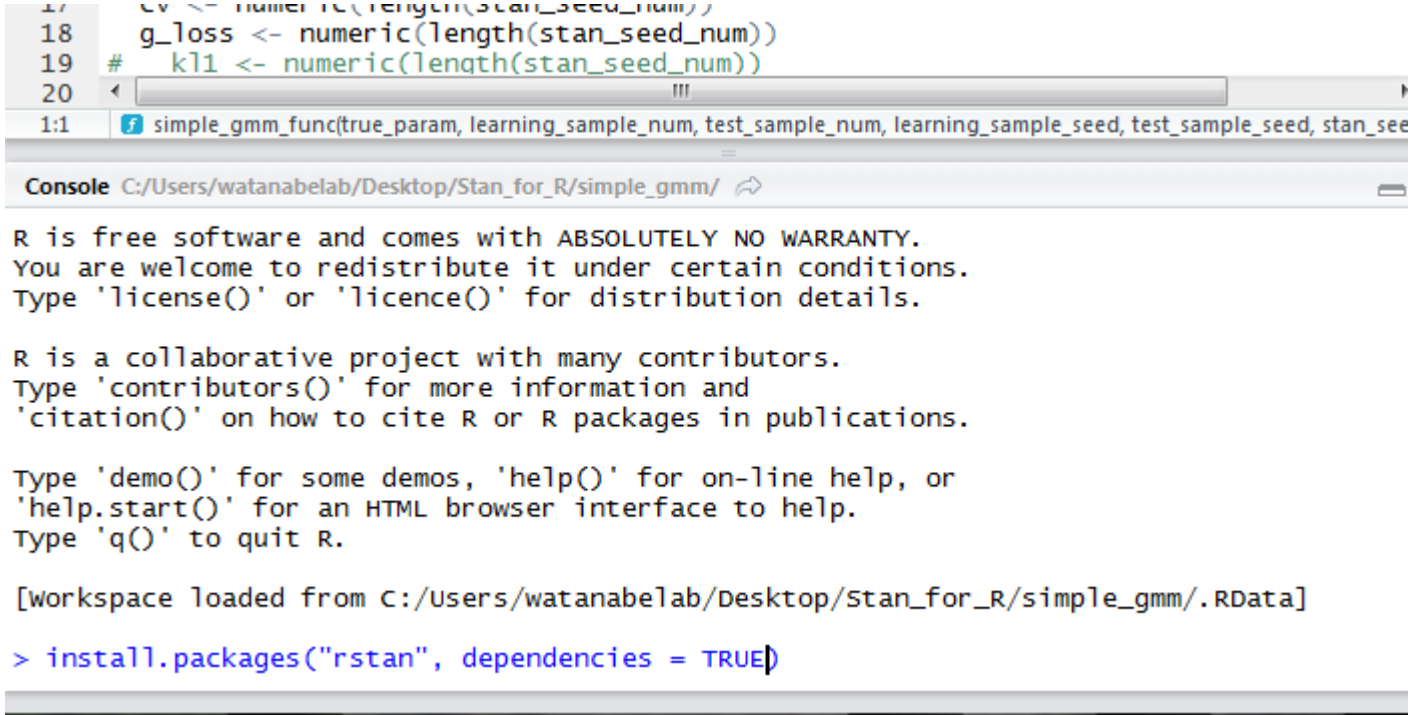
https://cran.r-project.org/bin/windows/Rtools/から最新バージョンをダウンロード

インストーラ
（丸の部分にチェックを入れる必要がある）



3．Stanのパッケージをインストール

Rを起動して install.package(“rstan”, dependencies=TRUE)
と入力して、Rを再起動



３．STANのコード記述方法

Stanのコードは7つのブロックからなる

１．functions{}:他のブロックで用いるユーザ定義の関数を記述する）

２．data{}:モデルに必要なデータやハイパーパラメータの型を宣言する

３．transformed data{}:データの中で宣言以外の処理をしたいものの宣言と処理を行う

4. parameters{}:サンプリングするパラメータの構造を宣言する

5. transformed parameters{}:パラメータの中で宣言以外の処理をするものの宣言と処理を行う

6. model{}:サンプリングしたい分布に対数を取ったものを記述する

7. generated quantities{}:各サンプリングで得られたパラメータ毎に計算することができるブロック

※$_1$modelブロック以外は省略可
※$_2$順番は1~7の順番で書く必要がある

データの型

int:整数型

real:実数型

real<lower=0,upper=1>:最小値0、最大値1の実数（他の型でも制約はつけることができる）

real a[N] : 変数aに実数の要素数がNの配列を宣言

vector[N]:N次元ベクトル(要素は実数)

simplex[N]:N次元ベクトルで総和が1

matrix[N,M]:N行M列の行列(要素は実数)

cov_matrix[M]:M行M列の分散共分散行列

など

4.STANによるサンプリングの例

1.ベルヌーイ分布

$$p(x|p) = p^x(1-p)^{1-x}, \varphi(p) \propto p^{\alpha-1}(1-p)^{\beta-1}$$

STANのコード

```
data{
  int<lower=0> n;
  int<lower=0, upper=1> x[n];
}
parameters{
  real<lower=0, upper=1> theta;
}
model{
  increment_log_prob(beta_log(theta, 1,1));
  for(i in 1:n)
    increment_log_prob(bernoulli_log(x[i], theta));
}
```

データとかハイパーパラメータとかの型宣言をするブロック

サンプリングするパラメータの型宣言をするブロック

$\log \varphi(w) + \log p(x^n|w)$を定義するブロック

Rのコード

```
library(rstan)
rstan_options(auto_write=TRUE)
options(mc.cores = parallel::detectCores())

n <- 100
true_theta <- 0.2
x <- numeric(n)
for(i in 1:n){
  if(runif(1) < true_theta ) x[i] <- 1
  else x[i] <- 0
}
```

```
learning_data <- list(n = n, x = x)
fit <- stan(file = "bernoulli.stan", data =
learning_data,
            iter = 2000, chains = 4)
print(fit)
traceplot(fit, warmup=T)
post_theta <- extract(fit, permuted=T)
plot(post_theta$theta, rep(0,
length(post_theta$theta)))
```

Rでstanを実行するための関数
file:stanコードのファイル名
data:stan上に渡すデータ
iter:合計繰り返し回数(デフォルトはiter/2がバーンイン）
chains: 初期値を変える回数

実行結果の例

```
> print(fit)
Inference for Stan model: bernoulli.
4 chains, each with iter=2000; warmup=1000; thin=1;
post-warmup draws per chain=1000, total post-warmup draws=4000.

        mean se_mean   sd   2.5%    25%    50%    75%  97.5% n_eff Rhat
theta   0.22    0.00 0.04   0.14   0.19   0.22   0.24   0.30  1311    1
lp__  -53.69    0.02 0.71 -55.72 -53.86 -53.41 -53.23 -53.18  1847    1

Samples were drawn using NUTS(diag_e) at Tue Oct 20 14:37:25 2015.
For each parameter, n_eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor on split chains (at
convergence, Rhat=1).
>
```

mean:サンプリングの平均 se_mean:標準誤差　sd:標準偏差 2.5〜97.5:分位点　n_eff:有効サンプルサイズ
Rhat:Gelman,Rubinの収束判定指標 lp__:対数事後分布の値

2.正規分布

$$p(x|w) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right), \varphi(\mu|\alpha) \propto \exp\left(-\frac{\mu^2}{2\times 100^2}\right), \varphi(\sigma^2|\beta_1,\beta_2) \propto (\sigma^2)^{-(\beta_1+1)}\exp\left(-\beta_2\frac{1}{\sigma^2}\right)$$

```
data{
  int<lower=1> n;
  vector[n] x;
}
transformed data{
  real<lower=0> alpha; //hyperparameter of center
  real<lower=0> beta1; //hyperparameter of variance
  real<lower=0> beta2; //hyperparameter of variance
  alpha <- 100;
  beta1 <- 5;
  beta2 <- 5;
}
parameters{
  real mu; //parameter of center
  real<lower=0> vari; //parameter of variance
}
model{
  mu ~ normal(0,alpha);
  vari ~ inv_gamma(beta1, beta2);
  x ~ normal(mu, sqrt(vari));
}
generated quantities{
  real sigma; //stardard deviation
  sigma <- sqrt(vari);
}
```

ハイパーパラメータを最初から決めているため、transformed dataブロックに記述

サンプリングステートメント（increment_log_prob(normal_log(…))と同じ）

3.線形回帰

$$p(y|x,w) = \frac{1}{\sqrt{2\pi}^{M}} \exp\left(-\frac{\|y-Ax\|^2}{2}\right), \varphi(A|\lambda) \propto \prod_{i,j} \exp\left(-\lambda\left|A_{ij}\right|\right)$$

```
data{
  int<lower=0> n; //number of samples
  int<lower=0> N; //dimension of x
  int<lower=0> M; //dimension of y
  matrix[n,N] x;
  matrix[n,M] y;
  real lambda; //hyperparameter of A
}
parameters{
  matrix[N,M] A;
}
transformed parameters{
  real<lower=0> squared_error;
  squared_error <- 0;
  for(i in 1:n){
   squared_error <- squared_error + dot_self(y[i]-x[i]*A);
  }
}
model{
  for(i in 1:N){
   for(j in 1:M){
     increment_log_prob(-lambda*fabs(A[i][j]));// for lasso
     // increment_log_prob(-lambda*pow(A[i][j],2)); //for ridge
   }
  }
  increment_log_prob(-squared_error);
}
```

4.混合正規分布(一番簡単なやつ)

$$p(x|w) = (1-a)N(x) + aN(x-b), \varphi(w) \propto a^{\phi-1}(1-a)^{\phi-1}\exp\left(-\frac{1}{2\times 100^2}b^2\right)$$

```
functions{
 real gmm_log(real x, vector ratio, vector mu){
  vector[rows(ratio)] sum_term;
  int K;

  K <- rows(ratio);
  for(k in 1:K){
   sum_term[k] <- log(ratio[k]) + normal_log(x, mu[k],1);
  }
  return log_sum_exp(sum_term);
 }
 real gmm_vector_log(vector x, vector ratio, vector mu){
  vector[rows(ratio)] sum_term;
  real log_model;
  int K;
  int n;

  K <- rows(ratio);
  n <- rows(x);
  log_model <- 0;

  for(i in 1:n){
   for(k in 1:K){
    sum_term[k] <- log(ratio[k]) + normal_log(x[i], mu[k],1);
   }
   log_model <- log_model + log_sum_exp(sum_term);
  }
  return log_model;
 }
}
data{
 int<lower=0> n; //number of samples
 vector[n] x;
 real<lower=0> phi; //hyperparameter for mixing ratio
}
transformed data{
 real<lower=0> beta; //hyperparameter for centers(unmodeled)
 beta <- 100;
}
parameters{
 simplex[2] ratio; //mixing ratio
 real mu; //center of component
}
model{
 vector[2] mu_dash;
 mu_dash[1] <- 0;
 mu_dash[2] <- mu;

 //priors
 ratio ~ beta(phi,phi);
 mu ~ normal(0,beta);

 for(i in 1:n){
  x[i] ~ gmm(ratio, mu_dash); //increment_log_prob(gmm_log(x[i], ...))と同じ
 }
}
```

５．結論

1.STANのインストール方法を紹介した

2.STANを用いた事後分布からのサンプリングについていくつかの分布を用いて紹介した

是非STANを使ってみてください

補足

ハミルトニアンモンテカルロ法について

確率分布$p(w|x^n) \propto \exp\left(-H(w)\right)$からサンプリング

1.$w^{(0)}$の初期値を決めて、$t = 0$とする

2.補助変数を$p \sim N(0,1)$で発生させる

3.$w'(0) = w^{(t)}, p'(0) = p, \varepsilon$を決めて次の漸化式を$w' = w'(L)$になるまで繰り返す

$$p'\left(\tau + \frac{1}{2}\right) = p'(\tau) - \frac{\epsilon}{2}\frac{\partial}{\partial w}H\left(w'^{(\tau)}\right), \qquad w'(\tau+1) = w'(\tau) + \epsilon p'\left(\tau + \frac{1}{2}\right), \qquad p'(\tau+1) = p'\left(\tau + \frac{1}{2}\right) - \frac{\epsilon}{2}\frac{\partial}{\partial w}H\left(w'(\tau+1)\right)$$

4.$\min\left(1, \exp\left(\mathcal{H}\left(w'(L), p'(L)\right) - \mathcal{H}\left(w^{(t)}, p^{(t)}\right)\right)\right)$の確率で$w^{(t+1)} = w'(L)$,そうでなければ$w^{(t+1)} = w^{(t)}$

$$\mathcal{H}(w,p) = H(w) + \frac{p^2}{2}$$

5. $t = t + 1$としてtが欲しいサンプルの個数でなければ2に戻る

※Lとϵをいい感じに決めてくれるものとしてNo-U-Turn Sampler(NUTS)があり、STANのデフォルトアルゴリズムはNUTSである

Stanの参考文献

・岩波データサイエンスVol.1（特集）ベイズ推論とMCMCのフリーソフト

（買ってないが、目次にStanを紹介したところがある）

・[特集] ベイズ推論とMCMCのフリーソフト のサポートページ（インストールの仕方が載ってる）

（https://sites.google.com/site/iwanamidatascience/vol1/support_tokushu）

・基礎からのベイズ統計学: ハミルトニアンモンテカルロ法による実践的入門

（付録にRstanの例が載っている）

・Bayesian Data Analysis

（付録にRstanの例が載っている、開発者の方が書かれた本なので上のものより詳しい）