Request for:

1. Bayesian optimization
2. Bayesian inversion
3. Markov Chain Monte Carlo sampling

Tutorial [in Japanese]: Bayesian optimization

http://d2mate.mdxes.iir.isct.ac.jp/D2MatE/?page=tutorial&id=data

Bayesian statistics
Bayesian inversion
(Bayesian regression)

Gaussian process
Bayesian optimization

Bayesian inversion and MCMC

Bayesian way of thinking:

A prior distribution is updated to a posterior distribution using data.

Bayes theorem: $P(a|D) \propto P(D|a)\, P(a)$

posterior $\propto$ likelihood $\times$ prior

MCMC: provides a practical way to perform integration in Bayesian analysis

A random-walk sampling method to collect samples from the posterior distribution.

Bayesian optimization:

Choose the next experiment/calculation efficiently.

Inverse problem:

Estimate model parameters a from observed data *D*.

Bayesian inversion:

Estimate not a single best parameter, but a probability distribution of parameters.

Result:

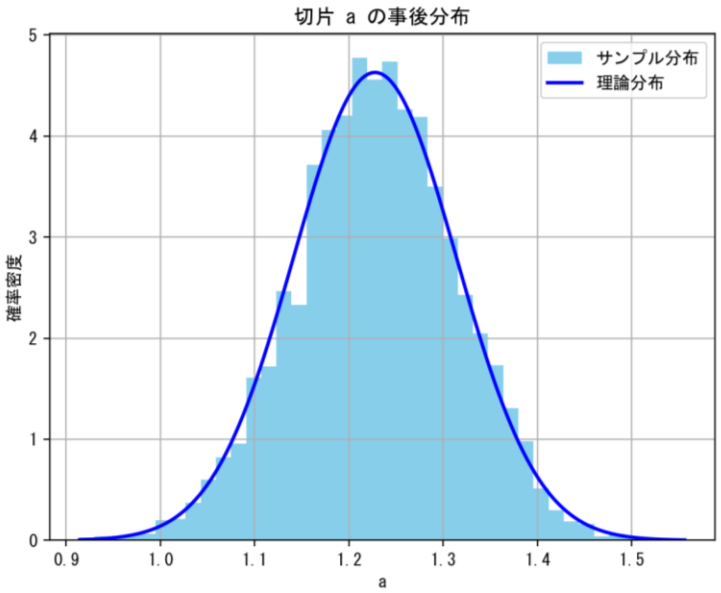
Mean value → estimated parameter

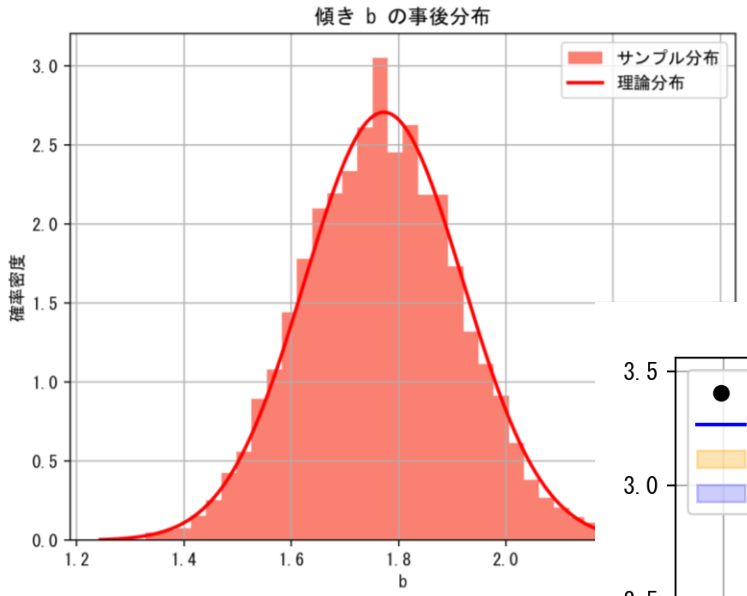
Distribution width → uncertainty of the parameter

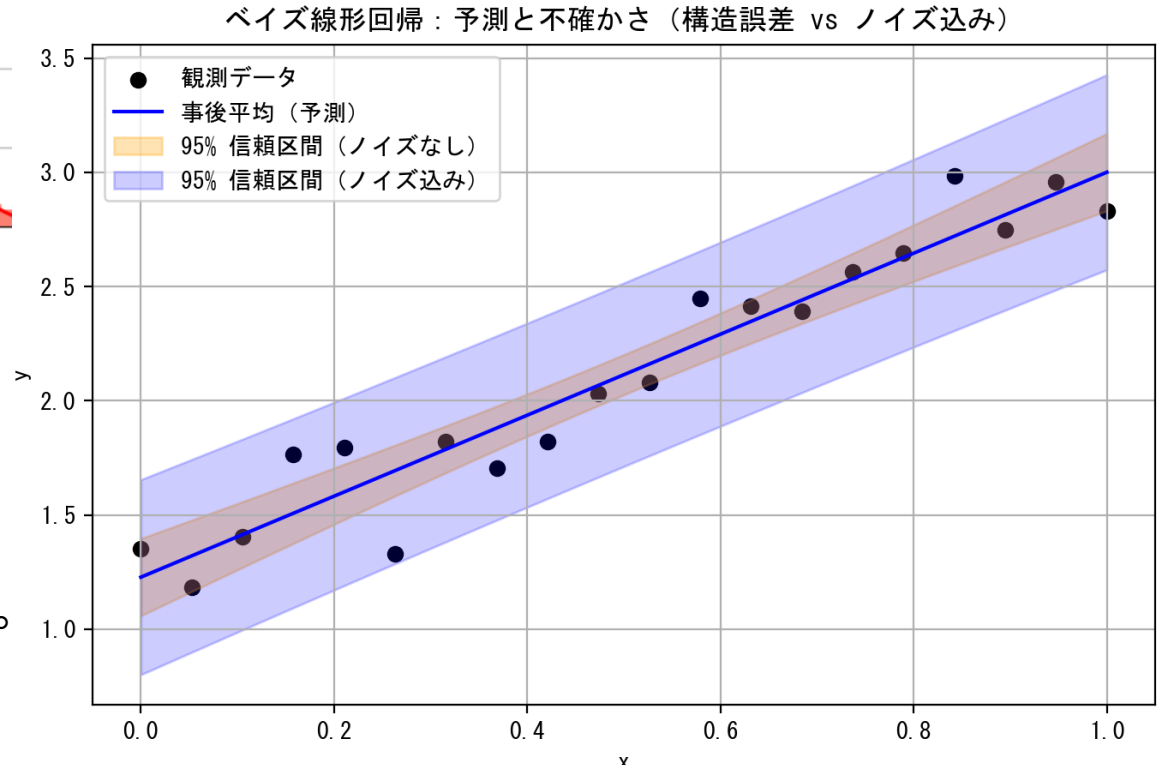
Bayes inversion for linear problem

bayesian_linear_regression_mcmc.py

Model function: y = a + bx







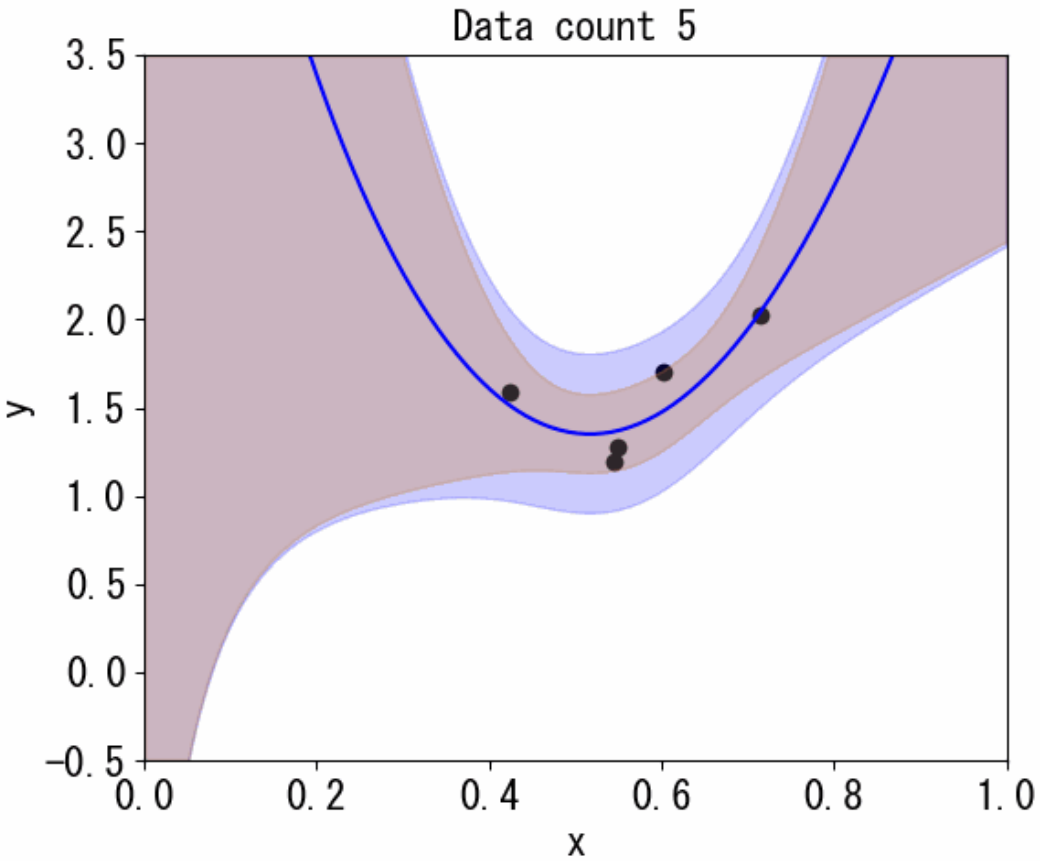
ノイズなし誤差：潜在関数（真の回帰関数）の不確かさ。
　パラメータの事後分布に基づく予測であり、観測ノイズは含まない。

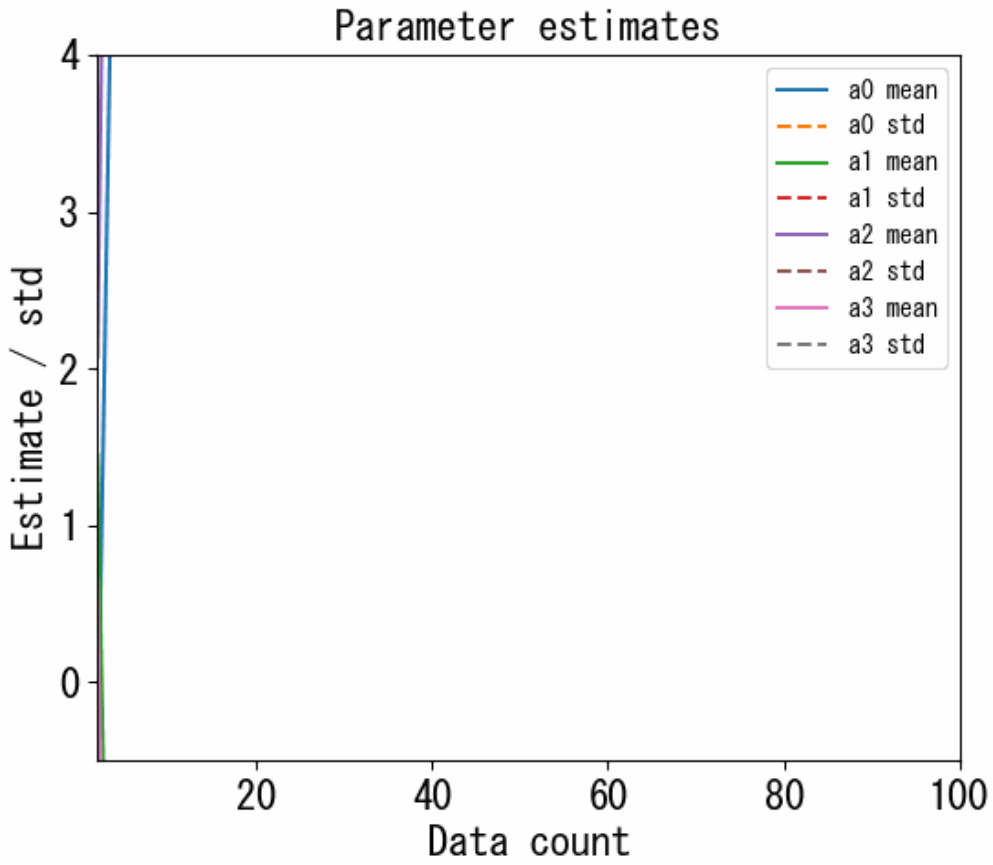
ノイズ込み誤差：観測値の予測分布の不確かさ。
　潜在関数の不確かさに加えて、観測ノイズのばらつきを含む。

Bayes inversion + bayes update

bayesian_linear_regression_mcmc_animation.py

Model function: $y = a0 + a1 * x + a2 * x^2 + a3 * \mathrm{sqrt}(x)$





Bayes optimization (using Gaussian process)

Bayes optimization demo (Gaussian process optimization)

> **python gp_simulation_animation.py ei 50 gp.gif**

Score:

ucb: Choose the point with the maximum upper confidence bound

$UCB(x) = \mu(x) + \kappa\sigma(x)$

an optimistic upper bound controlled by κ

ei: Choose the point with the maximum expected improvement

$EI(x) = \langle \max(f(x) - f_{best}), 0 \rangle$

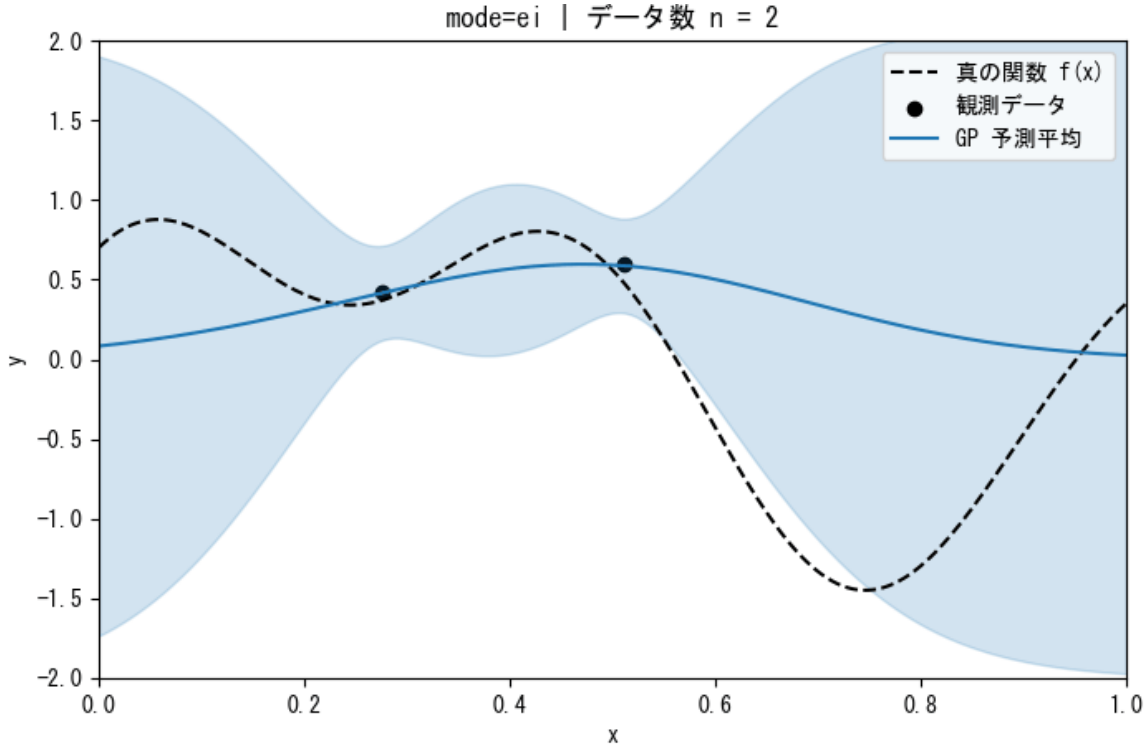
expected improvement over the current best

std: Choose the point with the maximum uncertainty

Reduce uncertainty over all data range

$\langle x \rangle, \mu(x)$: predicted mean of f(x)

$\sigma(x)$: predicted standard deviation



Bayes' Theorem and Materials Research

$$p(A, B) = p(B|A)p(A) = p(A|B)p(B)$$

$p(A, B)$	$p(B\|A)p(A)$		$p(A\|B)p(B)$	
A and B occur simultaneously	*probability of B for given A*	*Probability of A*	*probability of A for given A*	*Probability of B*

Bayes' theorem: $p(A|B) = p(B|A)p(A) \,/\, p(B)$

$p(A\|B)$	$p(B\|A)$	$p(A)$	$p(B)$
probability that A occurs after B		Prior probability	Marginal Probability (evidence)

From a material science perspective:

A: experimental condition B: measurement result then,

$p(B|A)$: probability of obtaining result *B* under condition *A*

$p(A|B)$: probability that condition *A* produced property *B*

By Bayes' theorem, the **condition *A*** that gives property ***B*** can be inferred from $p(A|B)$.

Problem: $p(B) = \Sigma_A \, p(B|A)$ requires performing all experiments.

An efficient sampling method is needed => **Monte Carlo sampling**.

ベイズの定理と材料研究

$$p(A,B) = p(B|A)p(A) = p(A|B)p(B)$$

$p(A,B)$: AとBが同時に生じる

$p(B|A)$: Aが生じた場合にBが生じる確率

$p(A)$: Aが生じる確率

ベイズの定理: $p(A|B) = p(B|A)p(A)/p(B)$

$p(B|A)$: B が生じた場合にAが生じる確率

$p(A)$: 事前確率

$p(B)$: 周辺確率(エビデンス)

A: 実験条件　B: 測定結果　と考えると、

$p(B|A)$: A の条件で実験を行った結果、B が得られる確率

$p(A|B)$: 物性B が得られたときに実験条件 Aである確率

ベイズの定理により、物性B が得られる実験条件 A を $p(A|B)$ から推定できる

問題: $p(B) = \sum_A p(B|A)$ を決めるためには全ての実験をやらないといけない

効率の良いサンプリング法が必要

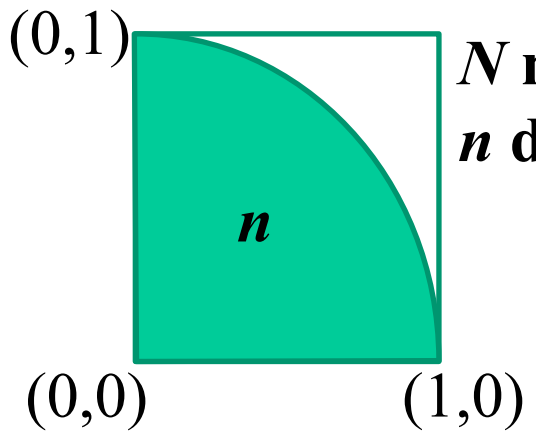
Monte Carlo simulation / sampling

Monte Carlo methods are used for various purposes:
for integration, stochastic simulations, optimization

- **Based on random number**
  How to generate random numbers in computer?
- **Application to multi-dimensional integration**
  Hit-and-miss Monte Carlo method
  Crude Monte Carlo method
- **Application to materials simulation**
  Metropolis Monte Carlo simulation
- **Integration in Bayesian statistics**

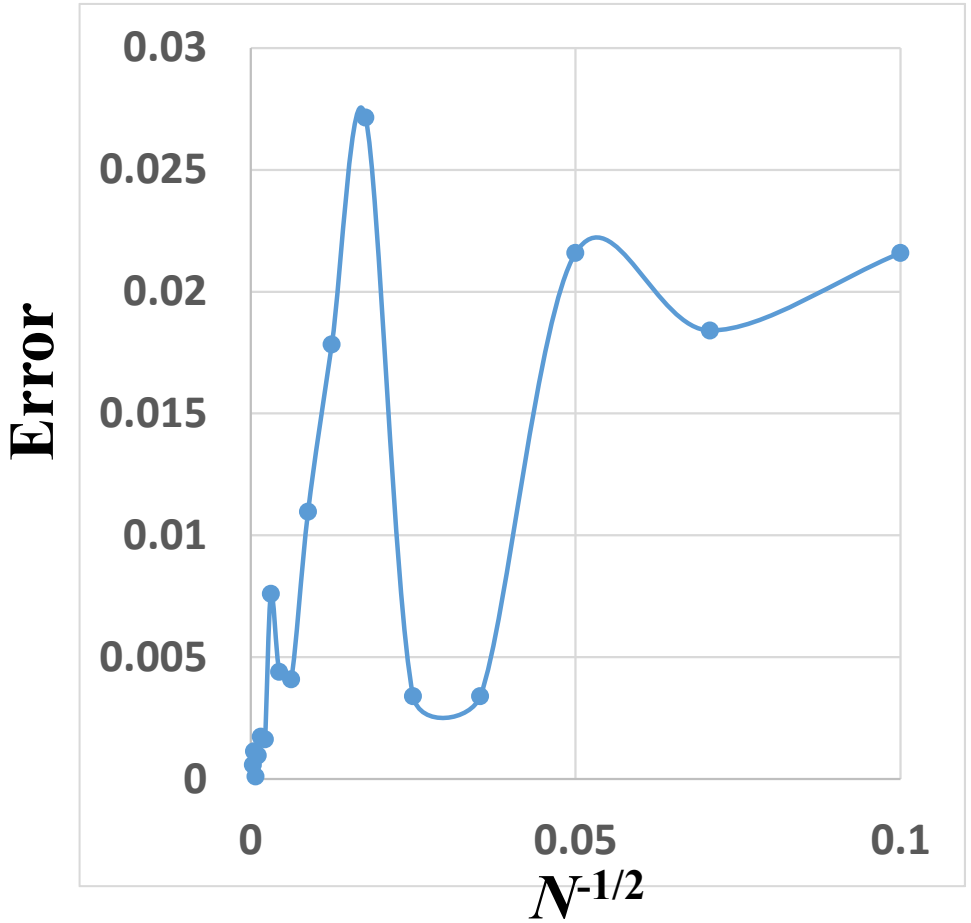
Integration: Hit-or-miss (試行錯誤的) Monte Carlo method

1. Generate random numbers (x, y) N times
2. Count the number that satisfies $(x^2 + y^2)^{1/2} < 1.0$ => n/N approaches the area of a quarter circle



N numbers distributes over the square
n drops in the quarter circle

N	$N^{-1/2}$	4S	|error|
100	0.1	3.12	0.021593
200	0.070711	3.16	0.018407
400	0.05	3.12	0.021593
800	0.035355	3.145	0.003407
1600	0.025	3.145	0.003407
3200	0.017678	3.16875	0.027157
6400	0.0125	3.12375	0.017843
12800	0.008839	3.130625	0.010968
25600	0.00625	3.1375	0.004093
51200	0.004419	3.137188	0.004405
102400	0.003125	3.133984	0.007608
204800	0.00221	3.139961	0.001632
409600	0.001563	3.139854	0.001739
819200	0.001105	3.14063	0.000963
1638400	0.000781	3.141702	0.000109
3276800	0.000552	3.14045	0.001142
6553600	0.000391	3.141	0.000593



Integration: Crude (基礎的) Monte Carlo method

Generate random numbers $0 \le r < 1$ N times

$S = \int_0^1 f(x)dx \sim \frac{1}{N}\sum_{i=1}^{N} f(x_i)$ is approximated

$$f(x) = 4\sqrt{1-x^2}$$

N	Hit-or-miss	crude
100	2.18E-01	1.23E-01
200	1.59E-03	1.31E-02
400	1.84E-02	5.53E-02
800	3.41E-03	2.41E-02
1600	2.16E-02	1.91E-02
3200	9.66E-03	1.70E-02
6400	1.15E-02	2.69E-03
12800	9.41E-03	1.11E-03
25600	3.47E-03	1.68E-03
51200	7.69E-03	1.83E-03
102400	2.57E-03	1.95E-03
204800	5.48E-03	2.52E-03
409600	2.93E-03	9.56E-04
819200	2.50E-03	7.10E-04
1638400	4.83E-04	4.01E-04
3276800	1.62E-05	8.08E-04
6553600	1.03E-03	3.59E-04

Numerical integration by random numbers

Good for multi-dimensional integration

***Ex*: Discrete Variational Xα method**

Integration: integ_montecarlo3d.py

Calculate the volume of radius 1.0 sphere: Exact value 4.188790205

Python integ_montecarlo3d.py

Output:

Hit-or-miss Monte-Carlo method

i	V	|error|
100	4.3200000000	0.13120979521360976
200	4.2000000000	0.011209795213609652
400	4.0200000000	0.16879020478639095
800	4.2000000000	0.011209795213609652
1600	4.2450000000	0.05620979521360958
3200	4.2025000000	0.013709795213609155
6400	4.1537500000	0.035040204786390916
12800	4.1868750000	0.001915204786390845
25600	4.1618750000	0.026915204786390312
51200	4.1620312500	0.026758954786390454
102400	4.1902343750	0.0014441702136096524
204800	4.1915625000	0.0027722952136093326
409600	4.1894921875	0.0007019827136094392
819200	4.1852148437	0.0035753610363906674
1638400	4.1913476562	0.002557451463609084
3276800	4.1906274414	0.0018372366198597945
6553600	4.1887829590	7.245802015276581e-06

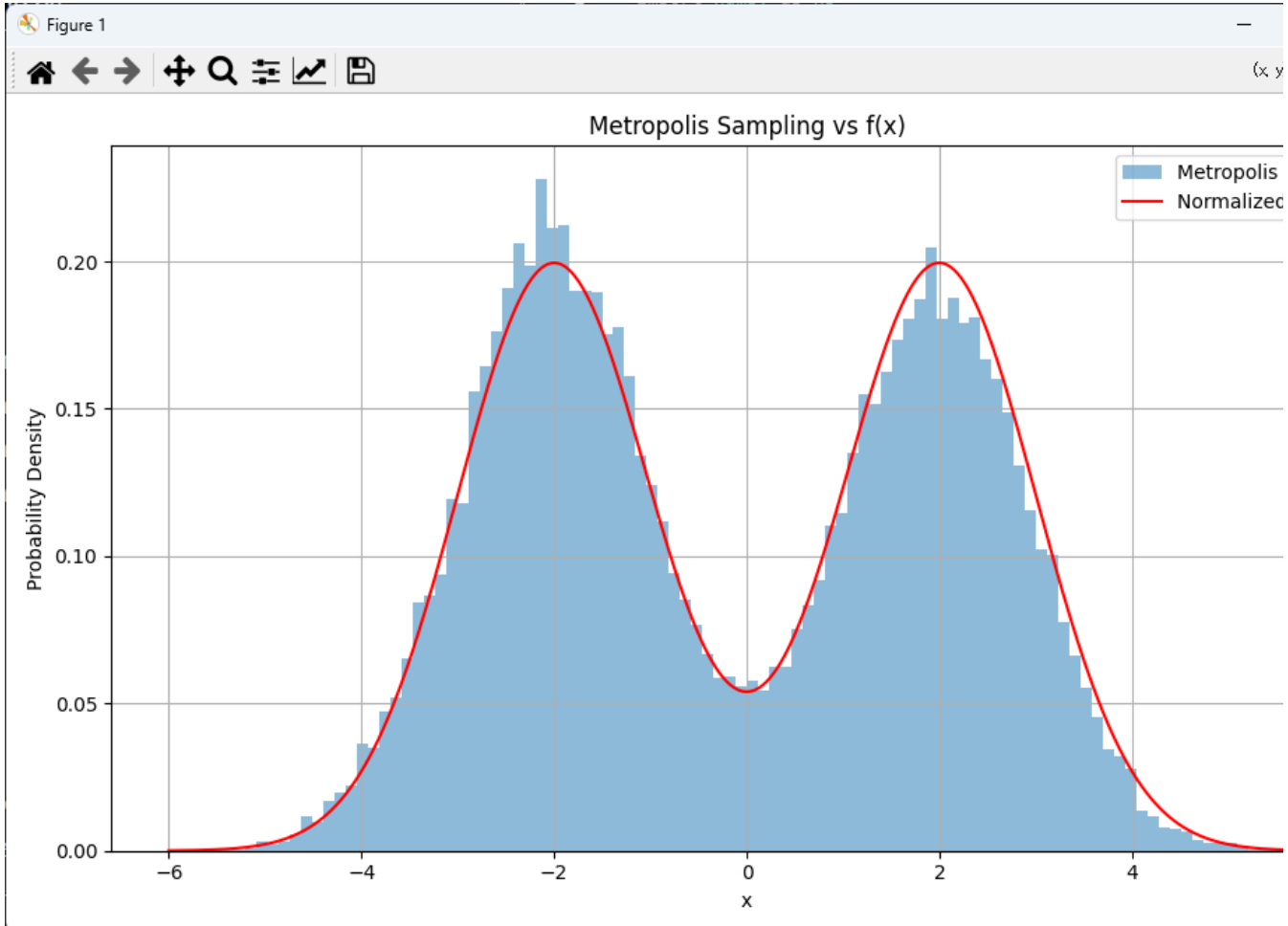
Error ∝ 1/N

Metropolis Monte Calro (MC) method: Generate arbitrary distributions

> python metropolis_rnd.py

method = ‘metropolis’

proposal_type = 'symmetric'



1. **Choose an initial point x_0.**
2. **For each step t, repeat the following:**
   - **From x_t, use the proposal distribution $q(x'|x)$ to generate $x_{proposal} \sim q(x'|x_t)$ q(x’|x)**
   - **Compute the acceptance probability $\alpha = \min(1, f(x\_proposal) / f(x\_t))$.**
   - **Generate a uniform random number u ~ [0,1].**
     If u < α, set $x_{t+1}=x_{proposal}$; otherwise, set $x_{t+1}=x_t$.
3. **If this is repeated sufficiently many times (including burn-in), the resulting sample sequence follows $p(x) \propto f(x)$.**

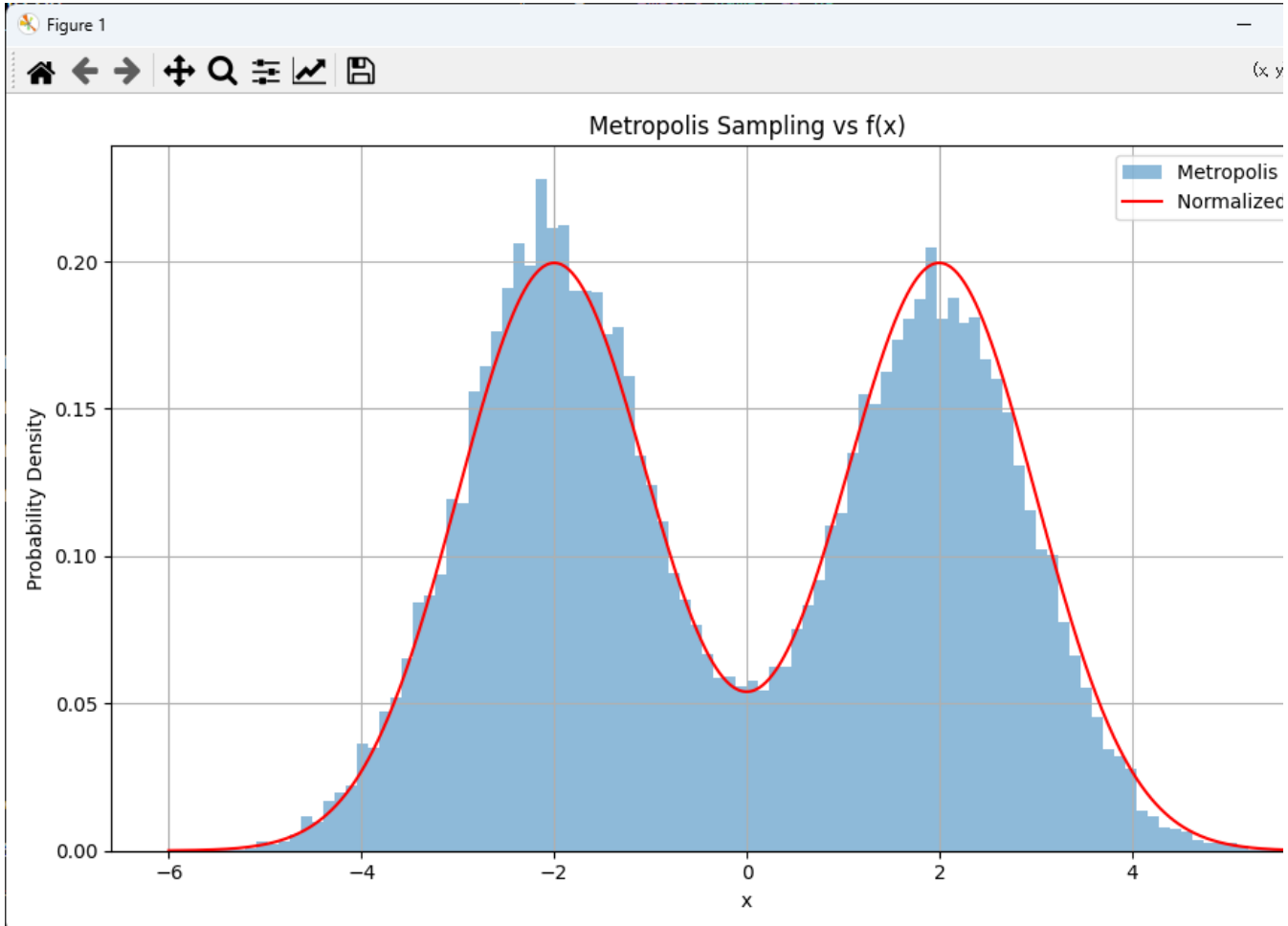
Because the result depends on the initial value, a burn-in period is required.

任意の確率密度関数 $f(x)$ に従う乱数: Metropolis Monte Calro (MC)法

> python metropolis_rnd.py

method = ‘metropolis’

proposal_type = 'symmetric'



1. 初期点 $x0$ を決める。
2. 各ステップ t で次を繰り返す:
   - x_t から提案分布$q(x'|x)$ を使って $x_{\text{proposal}}\sim q(x'|x_t)$ を生成。
   - 受容確率 $\alpha=\min(1, f(x_{\text{proposal}})/f(x_t))$ を計算。
   - 一様乱数 $u\sim[0,1]$ を生成し、$u<\alpha$ なら $x_{t+1}=x_{\text{proposal}}$、そうでなければ $x_{t+1}=x_t$ とする
3. これを十分な回数（バーンインも含め）繰り返すと、得られたサンプル列は $p(x)\propto f(x)$ に従う。

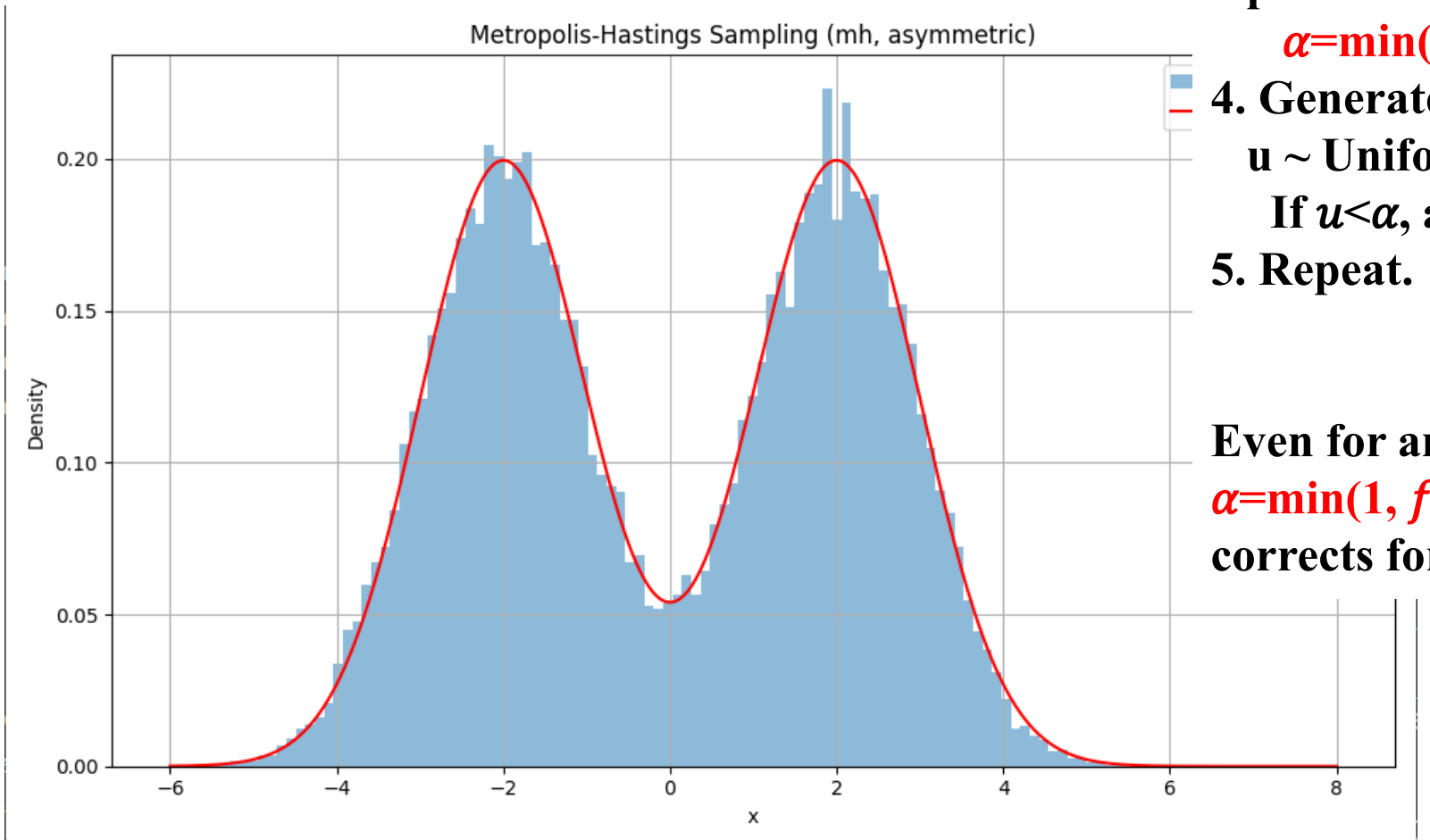
初期値に依存するので、バーンイン期間が必要

Improvement: Metropolis Hastings MC method

> python metropolis_rnd.py 5000 mh

method = ‘mh’

proposal_type = 'symmetric'



1. Choose an initial point x_0.

2. From the current point x, propose a new point $x' \sim q(x'|x)$.

3. Compute the following acceptance probability:

$$\alpha = \min\left(1, \frac{f(x') \cdot q(x|x')}{f(x) \cdot q(x'|x)}\right)$$

4. Generate a uniform random number u ~ Uniform(0,1).

If $u<\alpha$, accept x'; otherwise, keep x.

5. Repeat.

Even for an asymmetric proposal distribution,

$$\alpha = \min\left(1, \frac{f(x') \cdot q(x|x')}{f(x) \cdot q(x'|x)}\right)$$

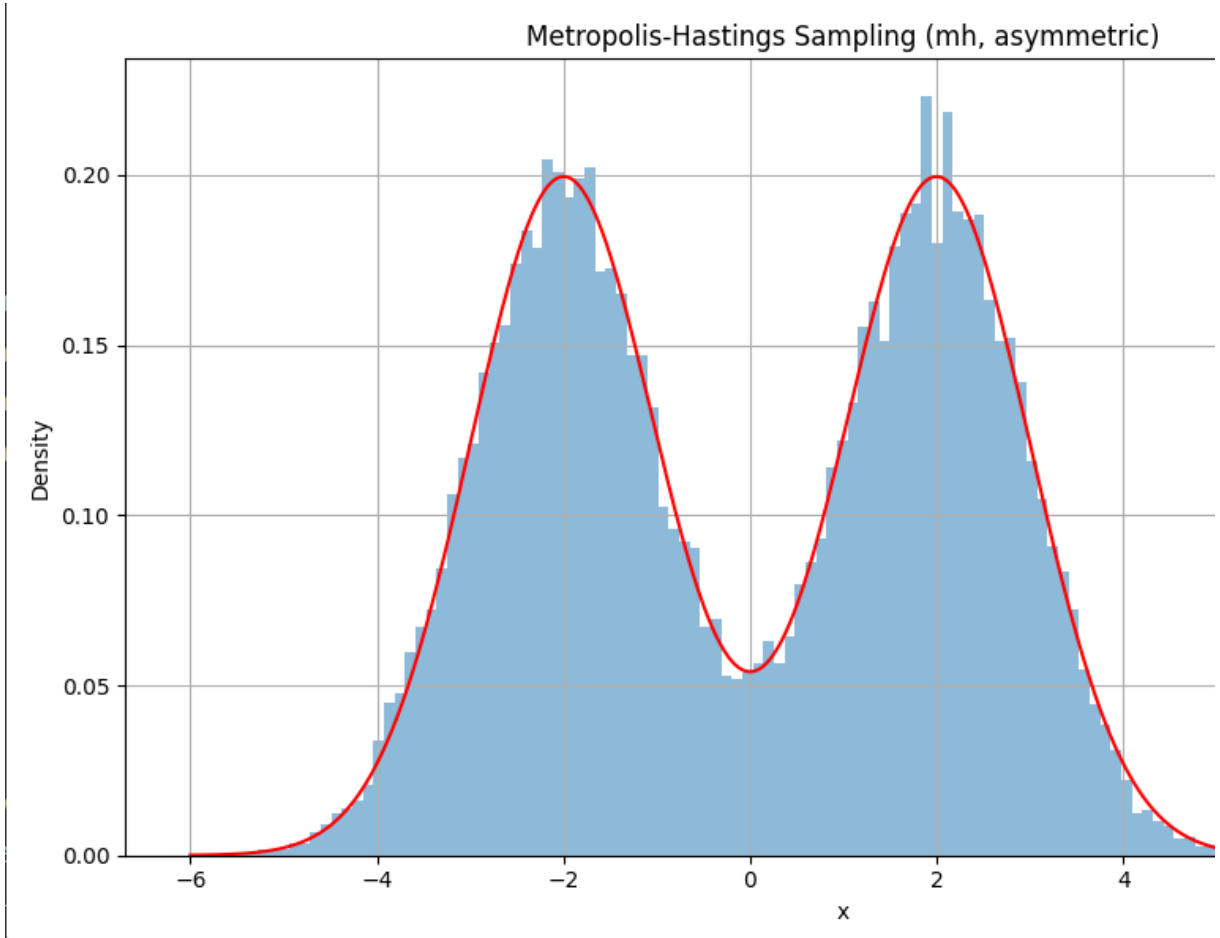
corrects for the asymmetry.

Improvement: Metropolis Hastings MC法

> python metropolis_rnd.py 5000 mh

method = ‘mh’

proposal_type = 'symmetric'



1. 初期点 x_0 を決定。
2. 現在の点 x から新しい点 $x' \sim q(x'|x)$ を提案。
3. 以下の 受容確率 を計算:
   $\alpha = \min(1, f(x') \cdot q(x|x') / [f(x) \cdot q(x'|x)])$
4. 一様乱数 $u \sim \text{Uniform}(0,1)$ を生成し、
   $u < \alpha$ なら x' を受け入れ、
   そうでなければ x を再度受け入れる。
5. 繰り返し。

非対称な分布でも

$\alpha = \min(1, f(x') \cdot q(x|x') / [f(x) \cdot q(x'|x)])$

によって補正される

Monte Carlo simulation for statistical physics

How to sample **an ensemble that follows canonical statistics**

$$P_i \propto \exp(-E_i/k_B T)$$

MCMC:

Consider a physical state and calculate its potential energy, U_1.
Generate another physical state using random numbers, and calculate U_2.

1. If $\Delta U = U_2 - U_1 \leq 0$, accept the new state unconditionally.
   $\Delta U = U_2 - U_1 \leq 0$ であれば、無条件にその状態を採択する
2. If $\Delta U > 0$, accept it with probability $\exp(-\Delta U / k_B T)$.
   $\Delta U > 0$ であれば、$\exp(-\Delta U / k_B T)$ の確率で採択する

   In step 2, generate a random number $0 \leq r < 1$.
   If $r \leq \exp(-\Delta U / k_B T)$, accept the state;
   otherwise, reject it and return to state 1.

The ensemble generated by this procedure coincides with the canonical statistics.
The statistical averages of quantities are obtained by taking averages of the physical quantities over this ensemble.

.

Appendix

Monte Carlo method

Monte Carlo simulation

Monte Carlo methods are used for various purposes:
for integration, stochastic simulations, optimization

- **Based on random number**
  How to generate random numbers in computer?
- **Application to multi-dimensional integration**
  Hit-and-miss Monte Carlo method
  Crude Monte Carlo method
- **Application to materials simulation**
  Metropolis Monte Carlo simulation
- **Integration in Bayesian statistics**

Uniform random and Pseudorandom numbers

- It is not easy to generate “random” phenomenon
  ＝＞ Generate pseudorandom numbers by algorism

- Product congruence method (乗積合同法): a, b, L are positive integers

  $N_1 = a$

  $N_2 = bN_1 \bmod L$

  $N_3 = bN_2 \bmod L$

  ・・・

  $N \bmod L$ is the remainder of N divided by L

  produces pseudorandom numbers N in $0 \leq N \leq L - 1$

- Mixed congruence method (混合合同法): a, b, L are positive integers

  $N_1 = a$

  $N_2 = bN_1 + c \bmod L$

  $N_3 = bN_2 + c \bmod L$

  ・・・

∗ NOTE: $N_k = N_m$ results in periodicity in the generated numbers

疑似乱数の検定

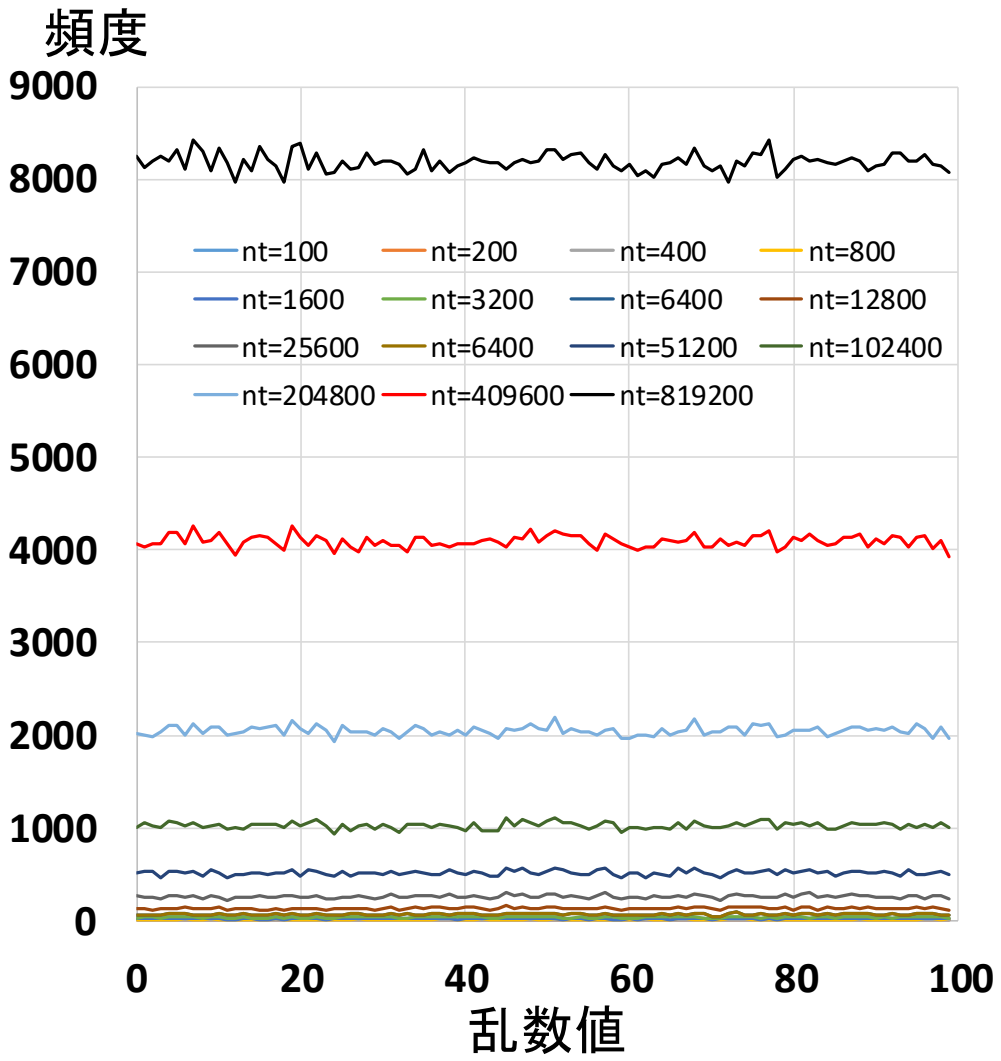
良い疑似乱数 (一様乱数に近い) の条件

- 分布が均一
- 周期性がない
- 標準偏差が $N^{1/2}$ に比例して増大

- **疑似乱数の発生の多くには “種 (seed)” が必要**
  - seedが同じなら乱数も同じになる。
  - 毎回seedを変える必要がある
    - (時計、乱数発生器など)
- **計算ごとに乱数が変わると困ることもある**
  - (デバッグ、計算結果の比較など)
    - seedを同じにして計算

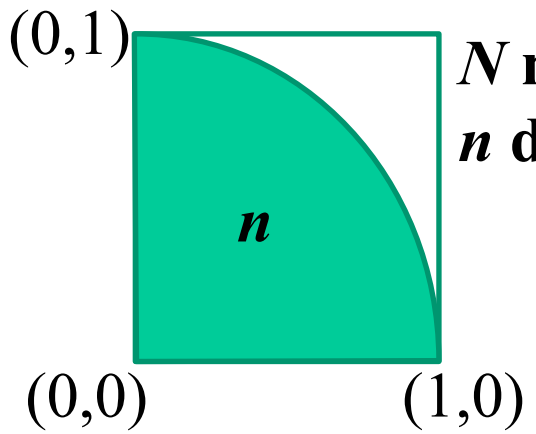
Perlの例 ＝＞

```
srand(0);
my @r;
for(my $i = 1 ; $i <= $n ; $i-
{
   $r[int(rand(100))]++;
}
```



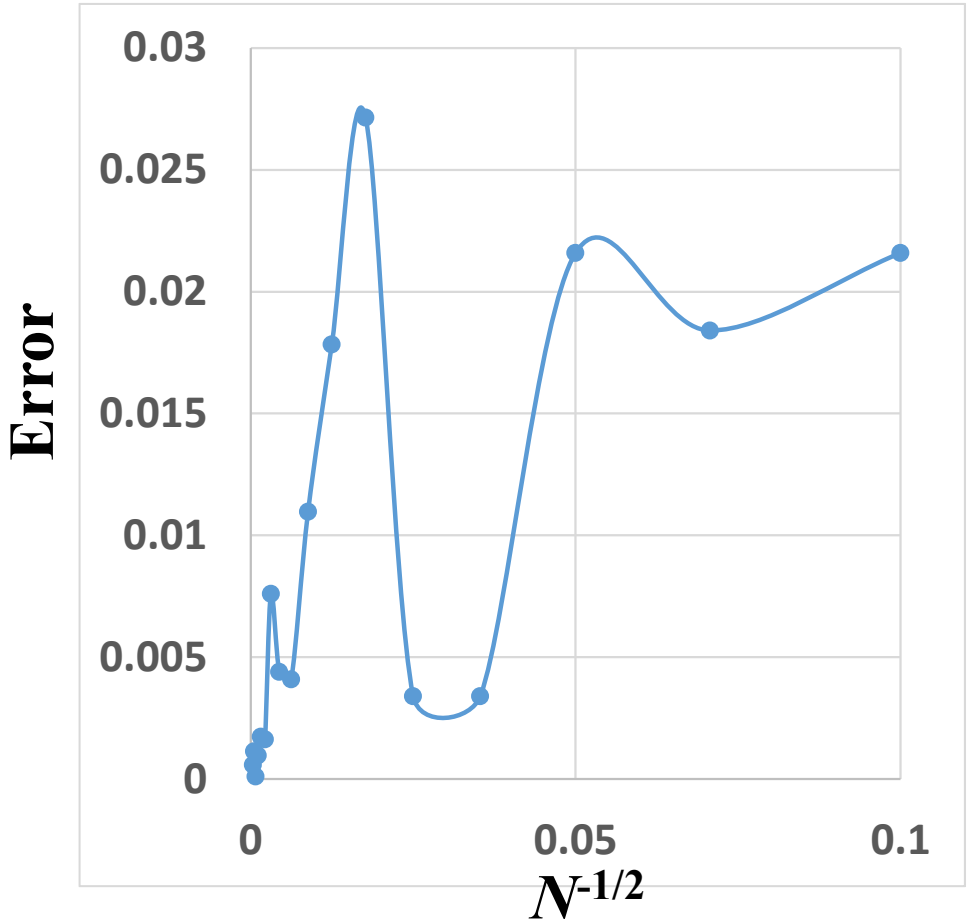
Hit-or-miss (試行錯誤的) Monte Carlo method

1. Generate random numbers (x, y) N times
2. Count the number that satisfies $(x^2 + y^2)^{1/2} < 1.0$ => n/N approaches the area of a quarter circle



N numbers distributes over the square
n drops in the quarter circle

N	$N^{-1/2}$	4S	|error|
100	0.1	3.12	0.021593
200	0.070711	3.16	0.018407
400	0.05	3.12	0.021593
800	0.035355	3.145	0.003407
1600	0.025	3.145	0.003407
3200	0.017678	3.16875	0.027157
6400	0.0125	3.12375	0.017843
12800	0.008839	3.130625	0.010968
25600	0.00625	3.1375	0.004093
51200	0.004419	3.137188	0.004405
102400	0.003125	3.133984	0.007608
204800	0.00221	3.139961	0.001632
409600	0.001563	3.139854	0.001739
819200	0.001105	3.14063	0.000963
1638400	0.000781	3.141702	0.000109
3276800	0.000552	3.14045	0.001142
6553600	0.000391	3.141	0.000593



Crude (基礎的) Monte Carlo method

Generate random numbers $0 \le r < 1$ N times

$S = \int_0^1 f(x)dx \sim \frac{1}{N}\sum_{i=1}^{N} f(x_i)$ is approximated

$$f(x) = 4\sqrt{1-x^2}$$

N	Hit-or-miss	crude
100	2.18E-01	1.23E-01
200	1.59E-03	1.31E-02
400	1.84E-02	5.53E-02
800	3.41E-03	2.41E-02
1600	2.16E-02	1.91E-02
3200	9.66E-03	1.70E-02
6400	1.15E-02	2.69E-03
12800	9.41E-03	1.11E-03
25600	3.47E-03	1.68E-03
51200	7.69E-03	1.83E-03
102400	2.57E-03	1.95E-03
204800	5.48E-03	2.52E-03
409600	2.93E-03	9.56E-04
819200	2.50E-03	7.10E-04
1638400	4.83E-04	4.01E-04
3276800	1.62E-05	8.08E-04
6553600	1.03E-03	3.59E-04

Numerical integration by random numbers

Good for multi-dimensional integration

***Ex*: Discrete Variational Xα method**

integ_montecarlo3d.py

Calculate the volume of radius 1.0 sphere: Exact value 4.188790205

Python integ_montecarlo3d.py

Output:

Hit-or-miss Monte-Carlo method

i	V	|error|
100	4.3200000000	0.13120979521360976
200	4.2000000000	0.011209795213609652
400	4.0200000000	0.16879020478639095
800	4.2000000000	0.011209795213609652
1600	4.2450000000	0.05620979521360958
3200	4.2025000000	0.013709795213609155
6400	4.1537500000	0.035040204786390916
12800	4.1868750000	0.0019152047863908 45
25600	4.1618750000	0.026915204786390312
51200	4.1620312500	0.026758954786390454
102400	4.1902343750	0.0014441702136096524
204800	4.1915625000	0.0027722952136093326
409600	4.1894921875	0.0007019827136094392
819200	4.1852148437	0.0035753610363906674
1638400	4.1913476562	0.002557451463609084
3276800	4.1906274414	0.0018372366198597945
6553600	4.1887829590	7.245802015276581e-06

Error ∝ 1/N

Random numbers that follows exponential distribution

http://www.sat.t.u-tokyo.ac.jp/~omi/random_variables_generation.html#Gauss

$$p(x;\lambda) = \lambda \exp(-\lambda x)$$ **(平均 1/λ, 分散 1/λ²)**

変換 $y = \exp(-x)$ を考えると、変換後の確率分布関数は

$$P(y) = P(x)\,|\,dx/dy\,|$$

となる。一様乱数 y から逆変換

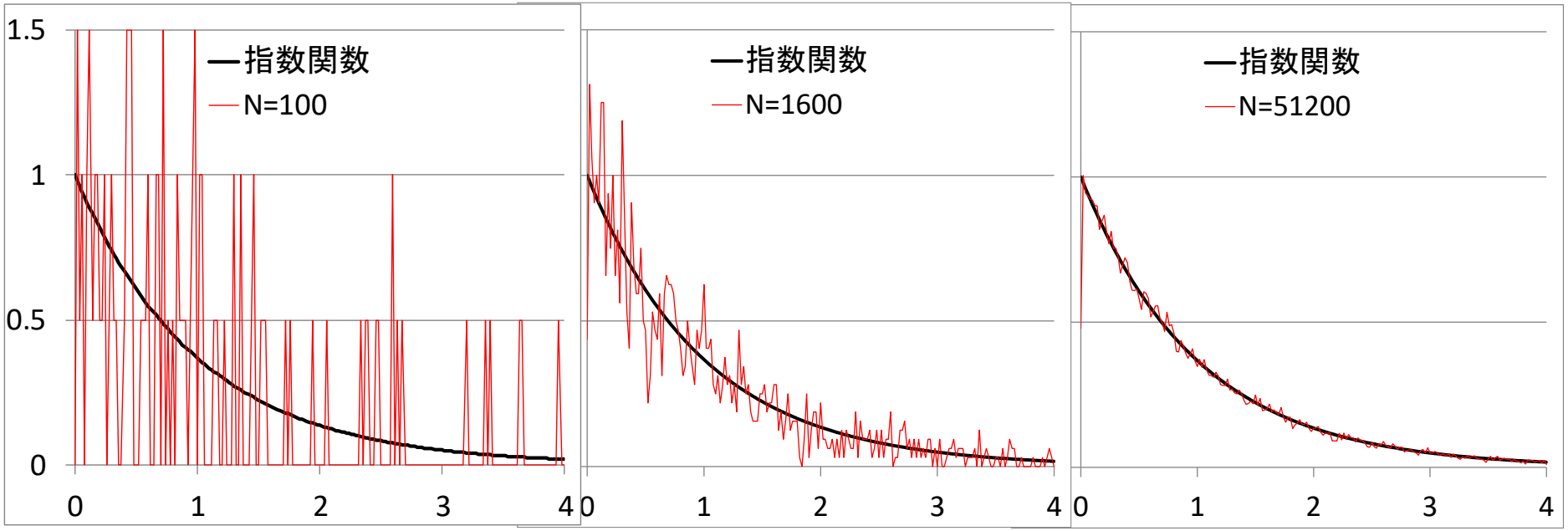
$$x = -\log(y)$$

により、λ = 1の指数分布に従う乱数が得られる。
任意の λ に対しては

$$x' = x / \lambda$$

にすればよい

Random numbers that follows exponential distribution



Random numbers that follows normal distribution (Box-Muller method)

$$p(x)=\left(\frac{1}{2\pi\sigma^2}\right)^{1/2}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$ (平均 μ, 分散 σ の正規分布)

一様乱数 x, y を作り、極座標へ変換

$$P(x,y)=P(x)P(y)=P(r,\theta)=\left(\frac{1}{2\pi}\right)r\exp\left(-\frac{r^2}{2}\right)$$

変数を r から r^2 に変える $P(r^2)=P(r)\,|\,dx/dy\,|=P(r)/(2r)$

$$P(r^2,\theta)=\left(\frac{1}{4\pi}\right)\exp\left(-\frac{r^2}{2}\right)$$

一様乱数 r, θ から

$x=r\cos(\theta), y=r\sin(\theta)$

が正規分布に従う乱数となるので、

$$z=\left(-2.0*\log(x)\right)^{1/2}*\sin(2\pi y)$$

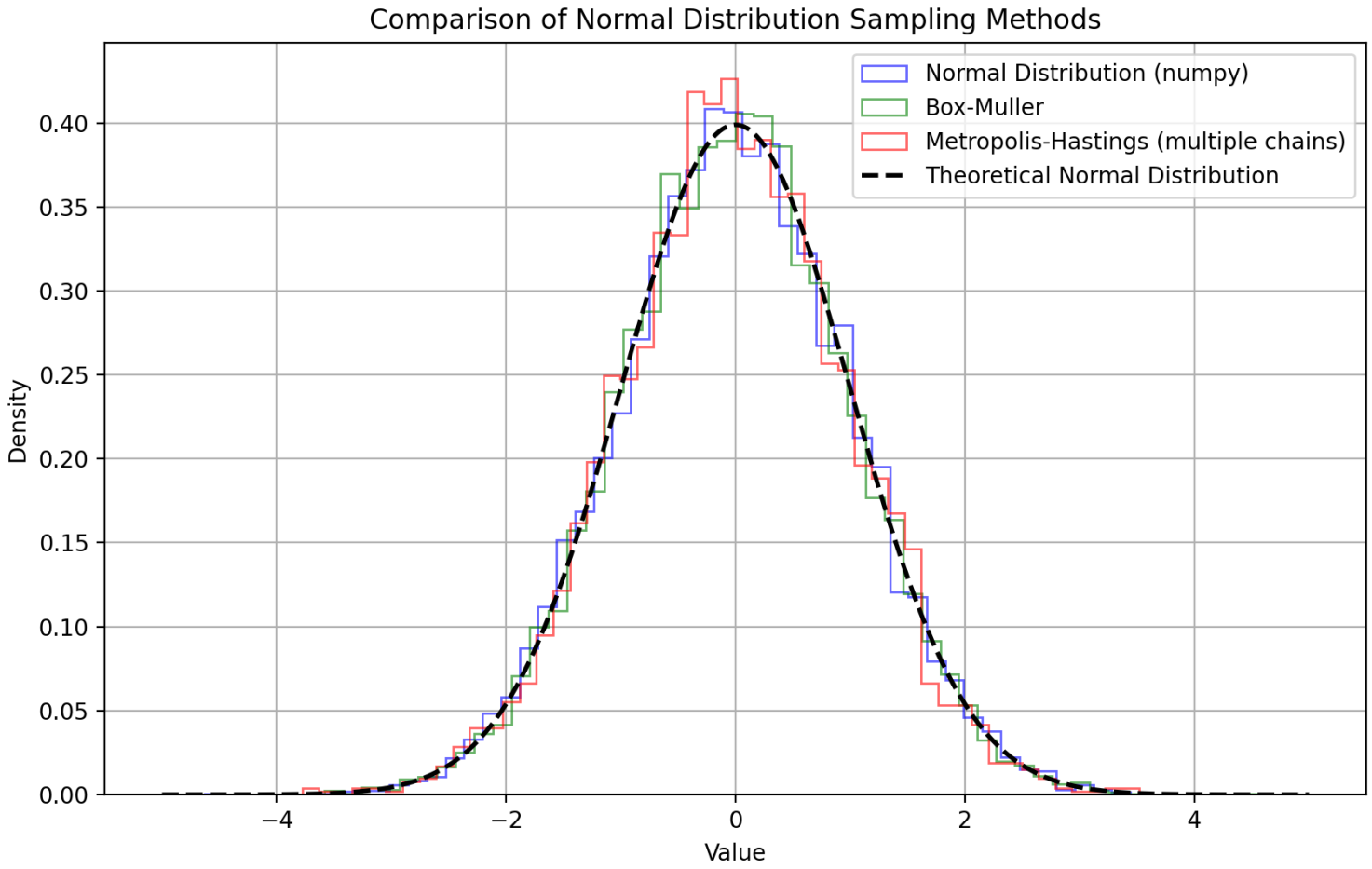
で計算できる。平均 μ, 分散 σ にするには

$z' = \mu + \sigma z$

にすればよい

正規分布乱数生成法の比較

normal_compare.py



計算時間

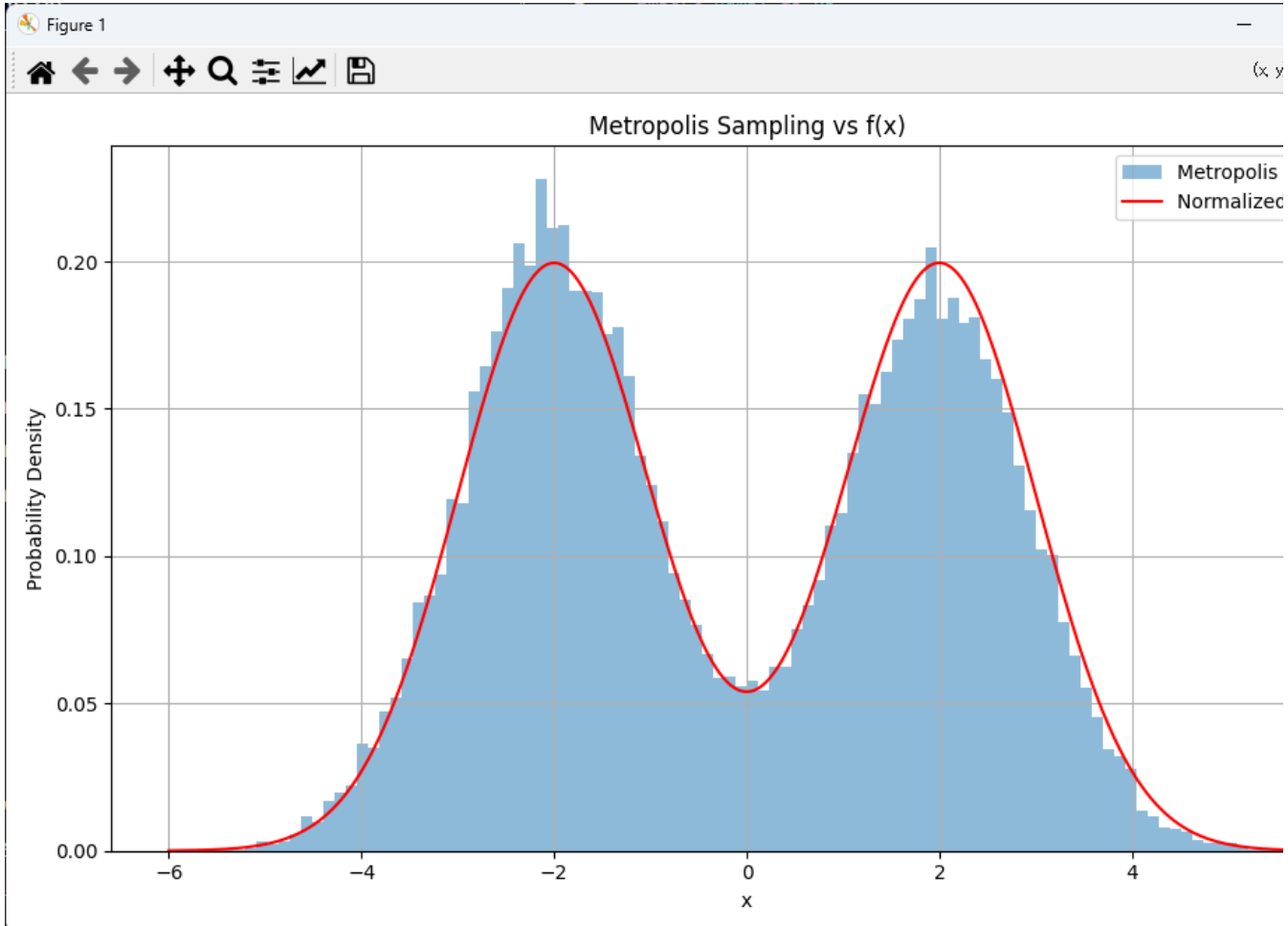
- numpy.random.normal: 0.000 ms
- Box-Muller法: 1.000 ms
- Metropolis-Hastings法: 138 ms

任意の確率密度関数 $f(x)$ に従う乱数: Metropolis Monte Calro (MC)法

metropolis_rnd.py で

method = ‘metropolis’

proposal_type = 'symmetric'



1. 初期点 $x0$ を決める。
2. 各ステップ t で次を繰り返す:
   - x_t から提案分布$q(x'|x)$ を使って $x_{\text{proposal}}\sim q(x'|x_t)$ を生成。
   - 受容確率 $\alpha=\min(1, f(x_{\text{proposal}})/f(x_t))$ を計算。
   - 一様乱数 $u\sim[0,1]$ を生成し、$u<\alpha$ なら $x_{t+1}=x_{\text{proposal}}$、そうでなければ $x_{t+1}=x_t$ とする
3. これを十分な回数（バーンインも含め）繰り返すと、得られたサンプル列は $p(x)\propto f(x)$ に従う。

初期値に依存するので、バーンイン期間が必要

Metropolis Monte Carlo法: Boltzmann分布

ある物理状態を考え、そのエネルギーを計算し U_1 とする。
乱数を使って別の物理状態を作り (マルコフ連鎖)、
そのエネルギーを U_2 とする。

1. $\Delta U = U_2 - U_1 \leq 0$ であれば、無条件にその状態を採択する
2. $\Delta U > 0$ であれば、$f(\Delta U) = \exp(-\Delta U/k_B T)$ の確率で採択する
   2. において、乱数 $0 \leq r \leq 1$ が $r \leq \exp(-\Delta U/k_B T)$ であれば採択、
   そうでなければ棄却し、状態1 をとりもどす

マルコフ連鎖により上記の採択・棄却条件で作られた集団は、
分布関数 $\exp(-\Delta U/k_B T)$ に収束する
この集団について平均をとれば物理量の統計的期待値が得られる。

Monte Carlo simulation for statistical physics

How to collect **an ensemble that follows**
canonical statistics $P_i \propto \exp(-E_i/k_BT)$

Metropolis Monte Carlo method

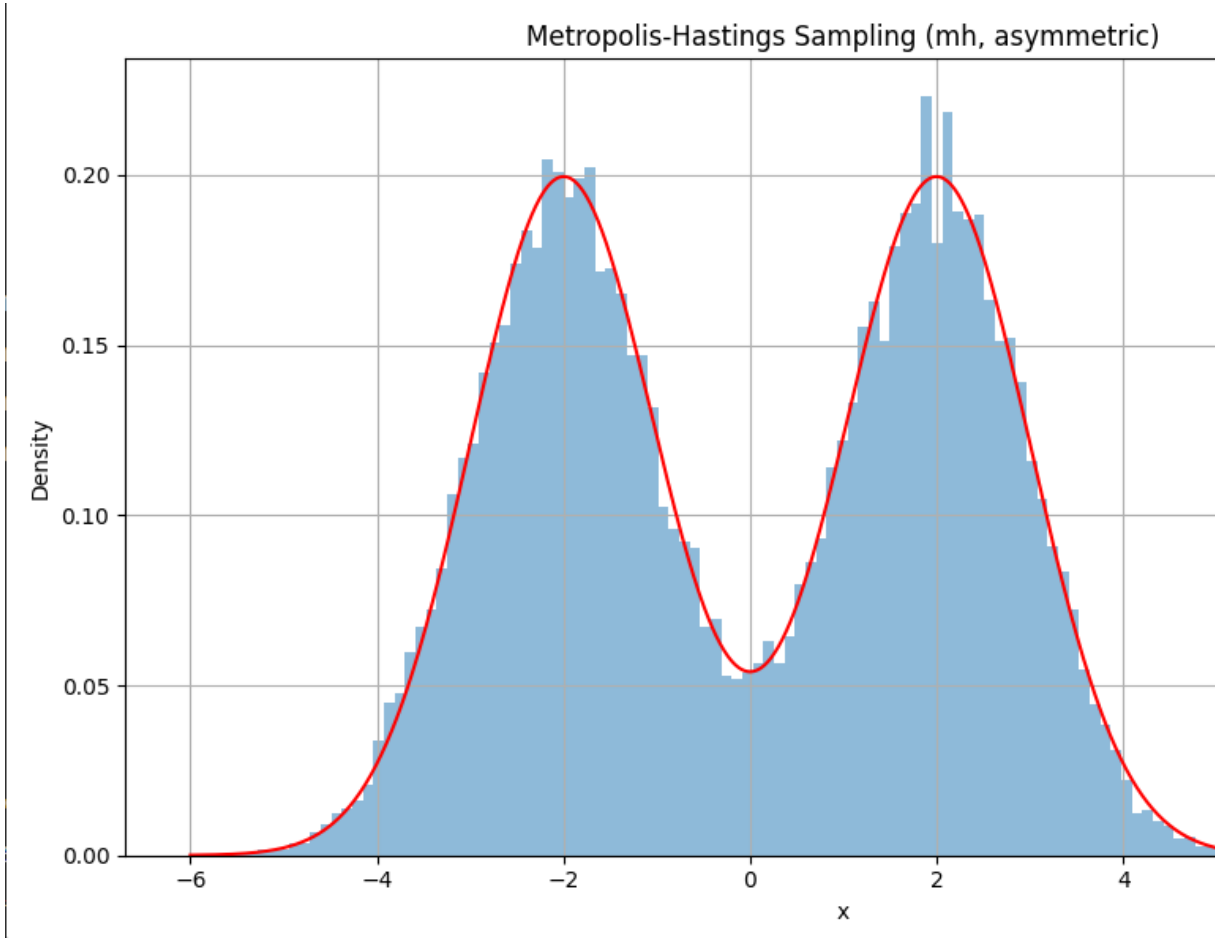
ある物理状態を考え、このポテンシャルエネルギーを計算し U_1 とする。
乱数を使って別の物理状態を作り、このポテンシャルエネルギーを U_2 とする。

1. $\Delta U = U_2 - U_1 \leq 0$ であれば、無条件にその状態を採択する
2. $\Delta U > 0$ であれば、$\exp(-\Delta U/k_BT)$ の確率で採択する
   2. において、乱数 $0 \leq r \leq 1$ が $r \leq \exp(-\Delta U/k_BT)$ であれば採択、
   そうでなければ棄却し、状態1 をとりもどす

という手続により作られた集団は、統計力学の母集団に一致する
この母集団について物理量の平均をとれば統計平均としての
物理量が得られる。

Metropolis Hastings MC法

metropolis_rnd.py で
method = ‘mh’



1. 初期点 x_0 を決定。
2. 現在の点 x から新しい点 $x' \sim q(x'|x)$ を提案。
3. 以下の 受容確率 を計算:
   $\alpha = \min(1, f(x') \cdot q(x|x') / [f(x) \cdot q(x'|x)])$
4. 一様乱数 $u \sim \text{Uniform}(0,1)$ を生成し、
   $u < \alpha$ なら x' を受け入れ、
   そうでなければ x を再度受け入れる。
5. 繰り返し。

非対称な分布でも
$\alpha = \min(1, f(x') \cdot q(x|x') / [f(x) \cdot q(x'|x)])$
によって補正される

Bayesian statistics

ベイズ統計学の特徴: 尤度関数

頻度主義的な考え方:

データ (x_k) に誤差が含まれており、パラメータ (母数) (a_k) は確定値

データ (x_k) が発生する確率:

確率密度関数: $P(x_i|a_k) = \prod_i \frac{1}{\sqrt{2\pi\sigma_i}} \exp\left[-\frac{\varepsilon_i(x_i|a_k)^2}{2{\sigma_i}^2}\right]$

$\varepsilon_i(x_i|a_k)$ は 誤差。$(x_i|a_k)$ は、x_iが確率変数でa_kが定数であることを示す。

ベイズ的な考え方:

データ (x_k) は与えられており確定値、(a_k) に不確かさ(確率分布) がある

(x_i) を定数とし、$P(a_i) = P(x_i|a_i)$が

パラメータ (母数) (a_k)がどれだけ尤もらしいか (尤度) を表す関数とみなす

=> 尤度関数と呼ぶ

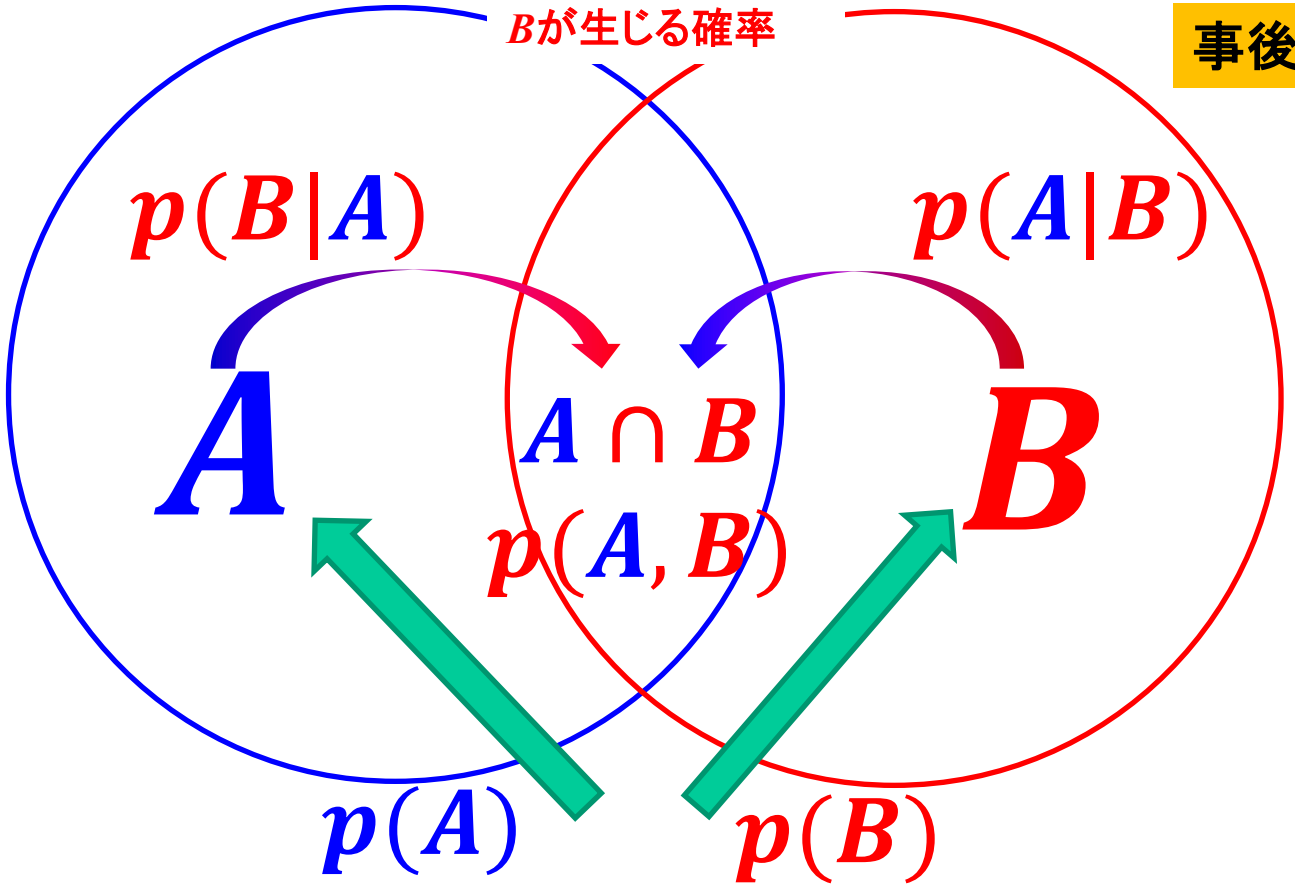
確率に関するベイズの定理

$$p(A,B) = p(B|A)p(A) = p(A|B)p(B)$$

同時確率　事後確率　事前確率

AとBが同時に生じる確率　Aが生じた場合にBが生じる確率　Aが生じる確率

事後確率 ＝ 条件付き確率



ベイズの定理: $p(A|B) = p(B|A)p(A)/p(B)$

事後確率　事前確率

ベイズの定理と材料研究

$$p(A,B) = p(B|A)p(A) = p(A|B)p(B)$$

$p(A,B)$: AとBが同時に生じる

$p(B|A)$: Aが生じた場合にBが生じる確率

$p(A)$: Aが生じる確率

ベイズの定理: $p(A|B) = p(B|A)p(A)/p(B)$

$p(B|A)$: B が生じた場合にAが生じる確率

$p(A)$: 事前確率

$p(B)$: 周辺確率(エビデンス)

A: 実験条件　B: 測定結果　と考えると、

$p(B|A)$: A の条件で実験を行った結果、B が得られる確率

$p(A|B)$: 物性B が得られたときに実験条件 Aである確率

ベイズの定理により、物性B が得られる実験条件 A を $p(A|B)$ から推定できる

問題: $p(B) = \sum_A p(B|A)$ を決めるためには全ての実験をやらないといけない

効率の良いサンプリング法が必要

回帰のベイズ的考え方

- データ (x_{k}) は与えられており確定値
- パラメータ (a_{k}) に不確かさ(確率分布) がある

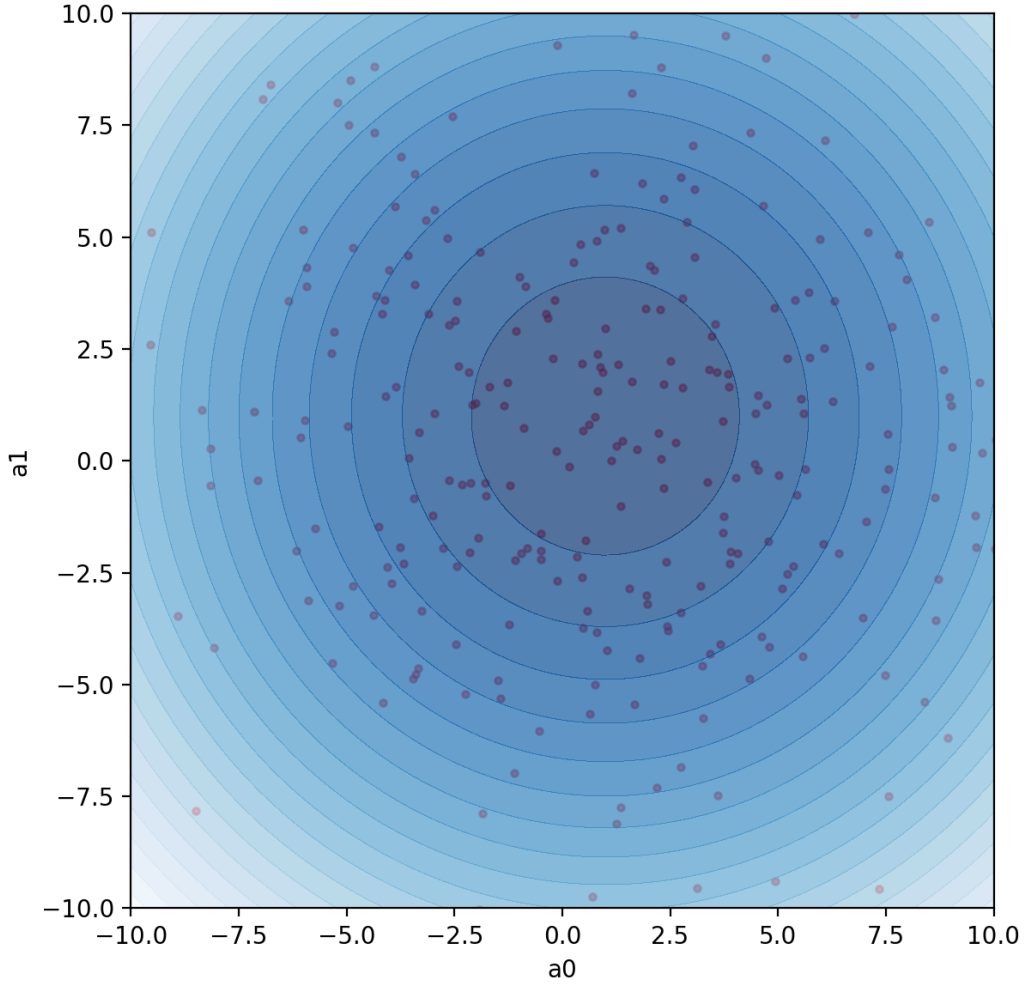
パラメータ a_i の分布とyの推定値の分布: 広い分布

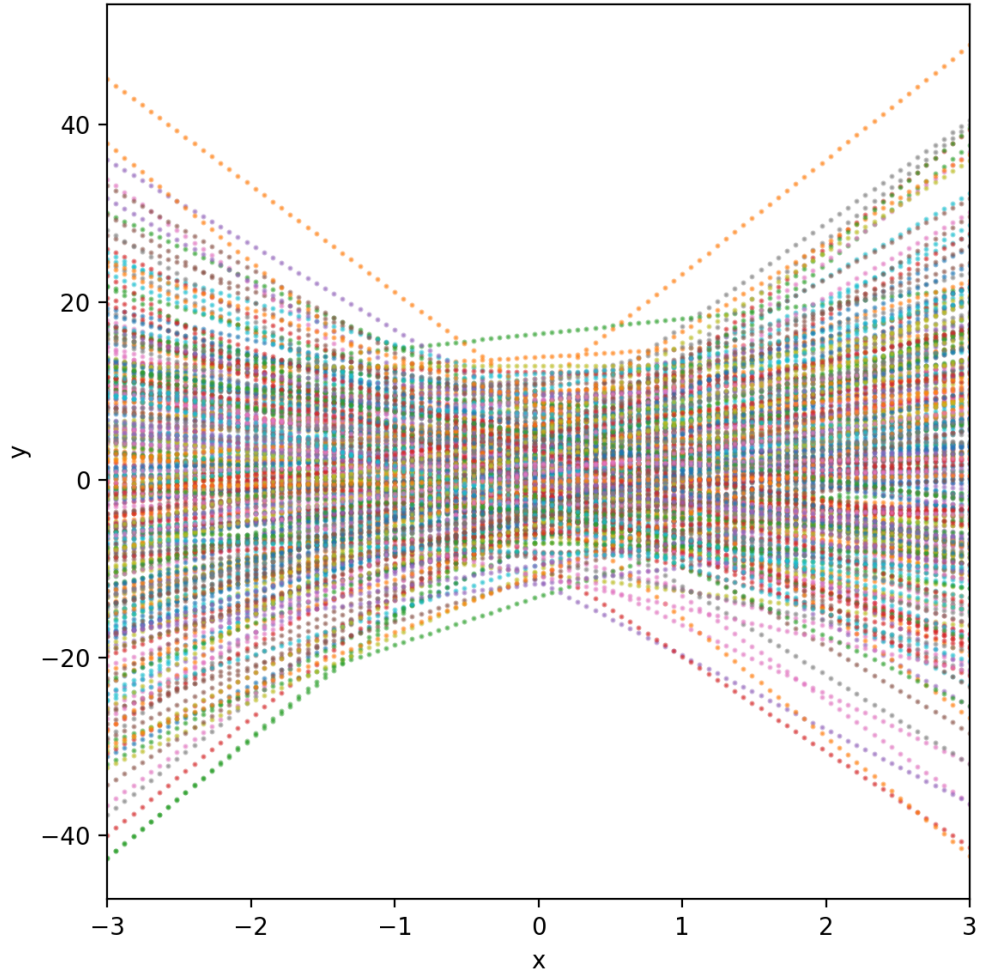
python visualize_bayesian_regression.py

y = a0 + a1 * x

a_i: 中心 (1, 1)、σ = 5.0 の2変数正規分布

マップの強度は対数を取っている





ベイズ的: データを学習して a_i の分布を狭める

1. **事前分布**: 最初は a に関する情報は無いことが多い
   適当な事前分布を仮定する **(主観確率)**:

$$p(\boldsymbol{a}) = \frac{1}{2\pi\sqrt{{\sigma_0}^2{\sigma_1}^2}} \exp\left(-\sum \frac{(a_i-\mu_i)^2}{2{\sigma_i}^2}\right) = N(\boldsymbol{a};\boldsymbol{\mu},\boldsymbol{\sigma})$$

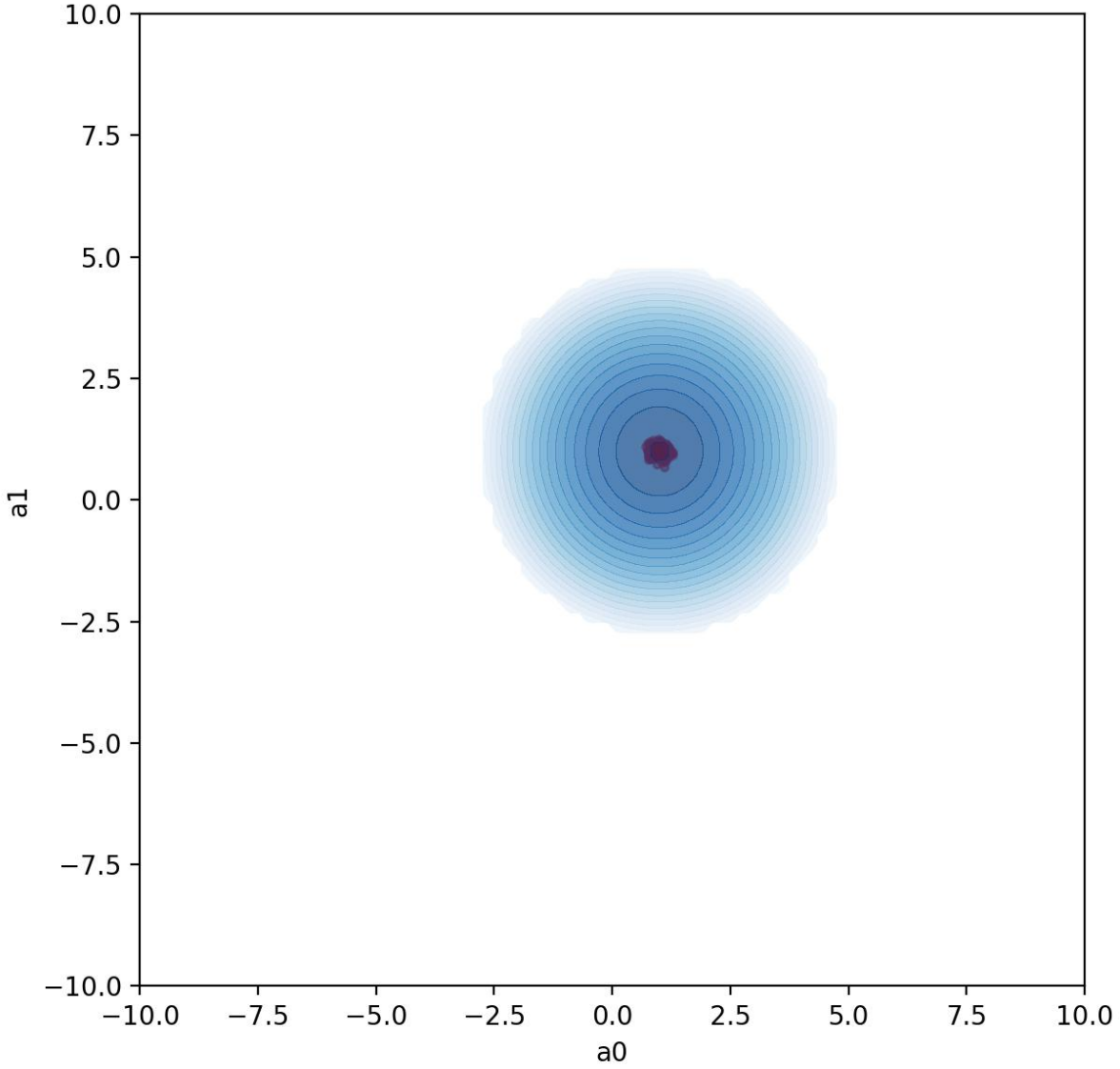
2. **データ $\boldsymbol{D} = (x_i, y_i)$ を学習**: a_i の範囲を狭めていく
   事後分布 (データ $\boldsymbol{D}$ が与えられた場合に
   $\boldsymbol{a}$ の確率密度分布がどうなるか):

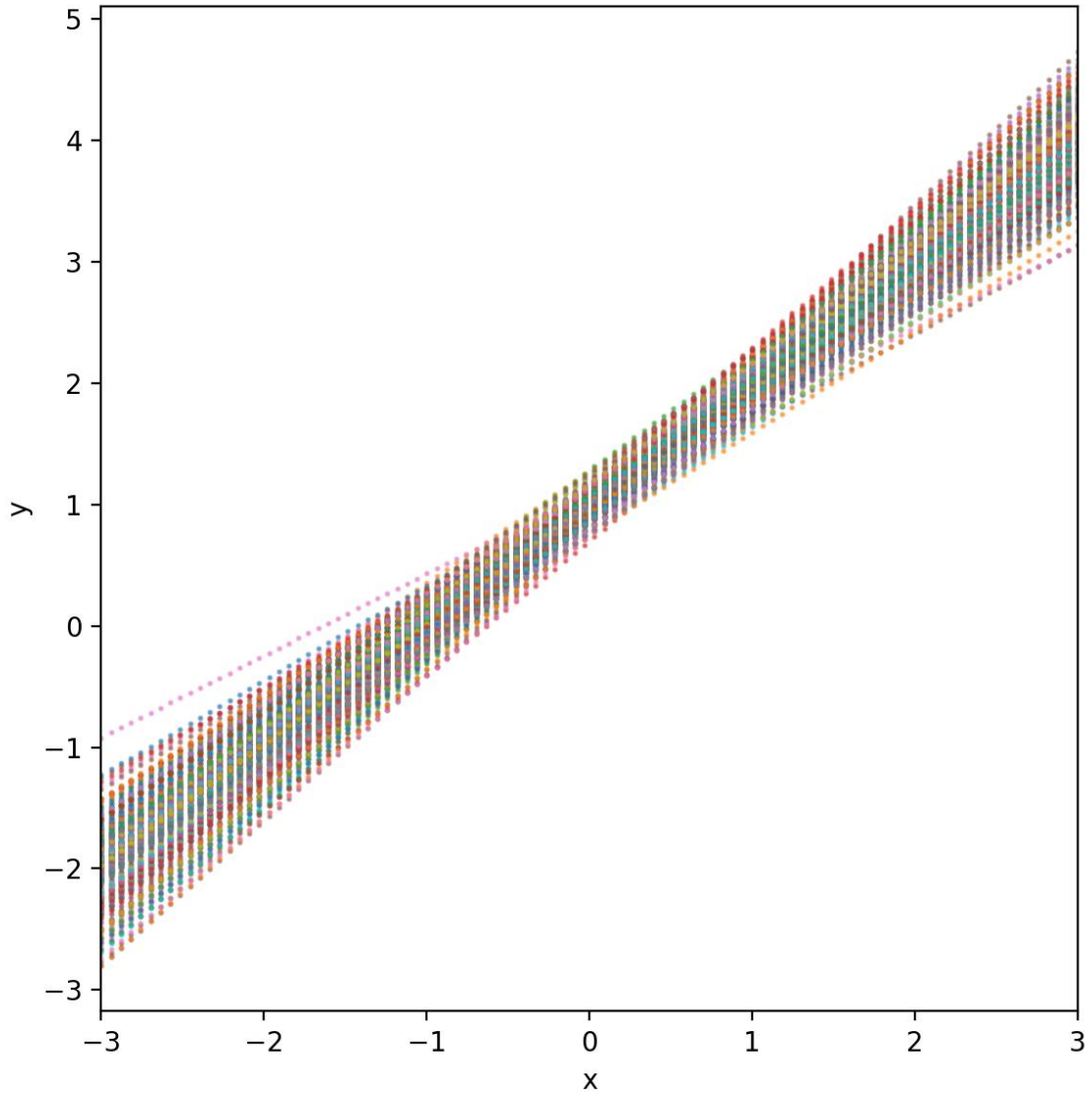
$$p(\boldsymbol{a}|\boldsymbol{D}) = \frac{1}{2\pi\sqrt{{\sigma_0}'^2{\sigma_1}'^2}} \exp\left(-\frac{(\boldsymbol{a}-\boldsymbol{\mu}')^2}{2\boldsymbol{\sigma}'^2}\right) = N(\boldsymbol{a};\boldsymbol{\mu}',\boldsymbol{\sigma}')$$

パラメータ a_i の分布とyの推定値の分布: 狭い分布

python visualize_bayesian_regression.py 1 1 0.1

a_i: 中心 (1, 1)、σ = 0.1の2変数正規分布





Bayes linear regression
as an example of Bayes inversion

・回帰は、モデル関数のパラメータを、モデル関数の実験値から逆に求める
“inversion problem”

ベイズ線形回帰

パラメータ数 p、データ数 n の場合:

設計行列 $X = \begin{pmatrix} x_{1,1} & \cdots & x_{p,1} \\ x_{1,2} & & x_{p,2} \\ \vdots & \ddots & \\ x_{1,n} & & x_{p,n} \end{pmatrix}$ $Y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$ $A = \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_p \end{pmatrix}$

モデル: $Y = XA + \varepsilon$

誤差ベクトル: $\varepsilon \sim N(0, \sigma^2 I)$ （平均 0、共分散行列 $\sigma^2 I$ の正規分布）

ベイズ線形回帰

モデル: $Y = XA + \varepsilon \;\; \varepsilon \sim N(0, \sigma^2 I)$ (σ^2 はRidge正則化係数に対応)

パラメータA の事前分布を仮定: $p(A) \sim N(A; \mu_0, \Sigma_0)$

尤度: $p(Y|A, X, \sigma^2) \sim N(Y; XA, \sigma^2 I)$

データXでAの分布を更新: 事後分布 (ベイズの定理)

(正規分布の積も正規分布になる)

$$p(A|Y, X, \sigma^2) = \frac{p(Y|A,X,\sigma^2)p(A)}{\int p(Y|A,X,\sigma^2)p(A)dA} \sim N(\hat{A}, \Sigma_A)$$

事後分布の共分散行列: $\Sigma_A = \left(\sigma^{-2} X^t X + {\Sigma_0}^{-1}\right)^{-1}$

期待値 : $\hat{A} = \Sigma_A\left(\sigma^{-2} X^t Y + {\Sigma_0}^{-1}\mu_0\right)$

Yの推定値Y_{new} の分布: $p(Y_{new}; X) = N(X_{new}\hat{A}, \sigma^2 + {X_{new}}^t \Sigma_A X_{new})$

ハイパーパラメータ $\sigma^2, \mu_0, \Sigma_0$などは最大事後確率推定 (MAP) により最適化

事後分布の導出

モデル: $Y = XA + \varepsilon \quad \varepsilon \sim N(0, \sigma^2 I)$

尤度: $p(Y|A) = \frac{1}{\sqrt{2\pi\sigma^2}^n} \exp\left(-\frac{1}{2\sigma^2}(Y - XA)^t(Y - XA)\right)$

事前分布: $p(A) = \frac{1}{\sqrt{2\pi}^n |\Sigma_0|^{1/2}} \exp\left(-\frac{1}{2}(A - \mu_0)^T {\Sigma_0}^{-1}(A - \mu_0)\right)$

$$\log p(A|Y) = \text{const} - p(Y|A)p(A)$$

$$= \text{const} - \frac{1}{2\sigma^2}(Y - XA)^t(Y - XA) - \frac{1}{2}(A - \mu_0)^T {\Sigma_0}^{-1}(A - \mu_0)$$

$$\sigma^{-2}(Y - XA)^t(Y - XA) = A^t\sigma^{-2}X^tXA - 2A^t\sigma^{-2}X^tY + \sigma^{-2}Y^tY$$

$$(A - \mu_0)^T {\Sigma_0}^{-1}(A - \mu_0) = A^t {\Sigma_0}^{-1} A \quad - 2A^t {\Sigma_0}^{-1}\mu_0 + {\mu_0}^t {\Sigma_0}^{-1}\mu_0$$

$$\log p(A|Y) = \text{const} - \frac{1}{2}A^t\left(\sigma^{-2}X^tX + {\Sigma_0}^{-1}\right)A + A^t\left(\sigma^{-2}X^tY + {\Sigma_0}^{-1}\mu_0\right)$$

$$= \text{const} - \frac{1}{2}\left(A - \hat{A}\right)^t {\Sigma_A}^{-1}\left(A - \hat{A}\right)$$

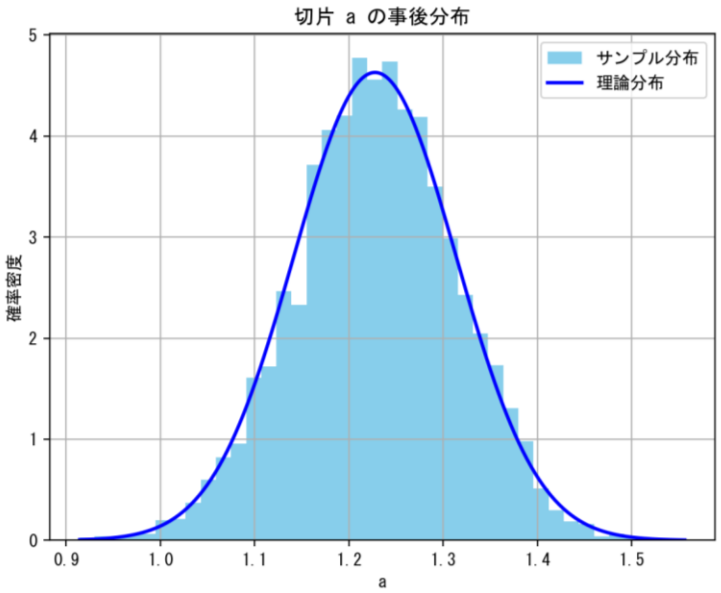
※ $\boldsymbol{{\Sigma_A}^{-1} = \left(\sigma^{-2}X^tX + {\Sigma_0}^{-1}\right)}$

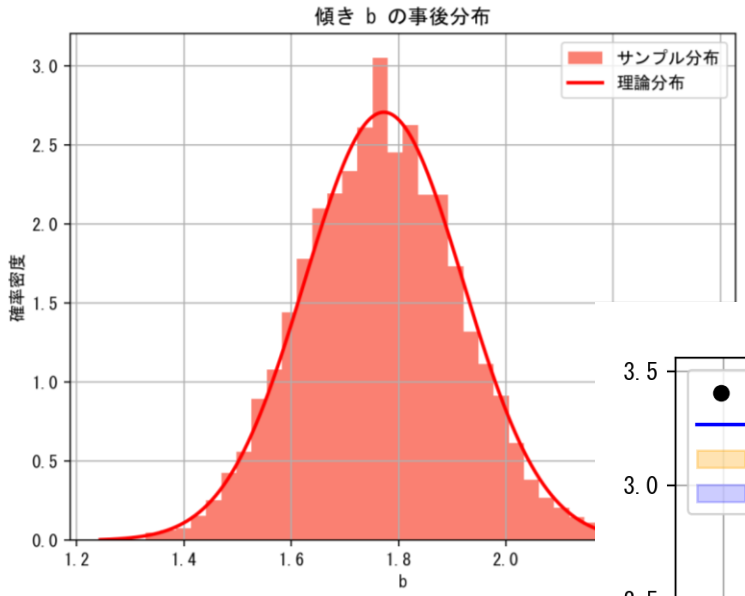
$\boldsymbol{\hat{A} = \Sigma_A\left(\sigma^{-2}X^tY + {\Sigma_0}^{-1}\mu_0\right)}$

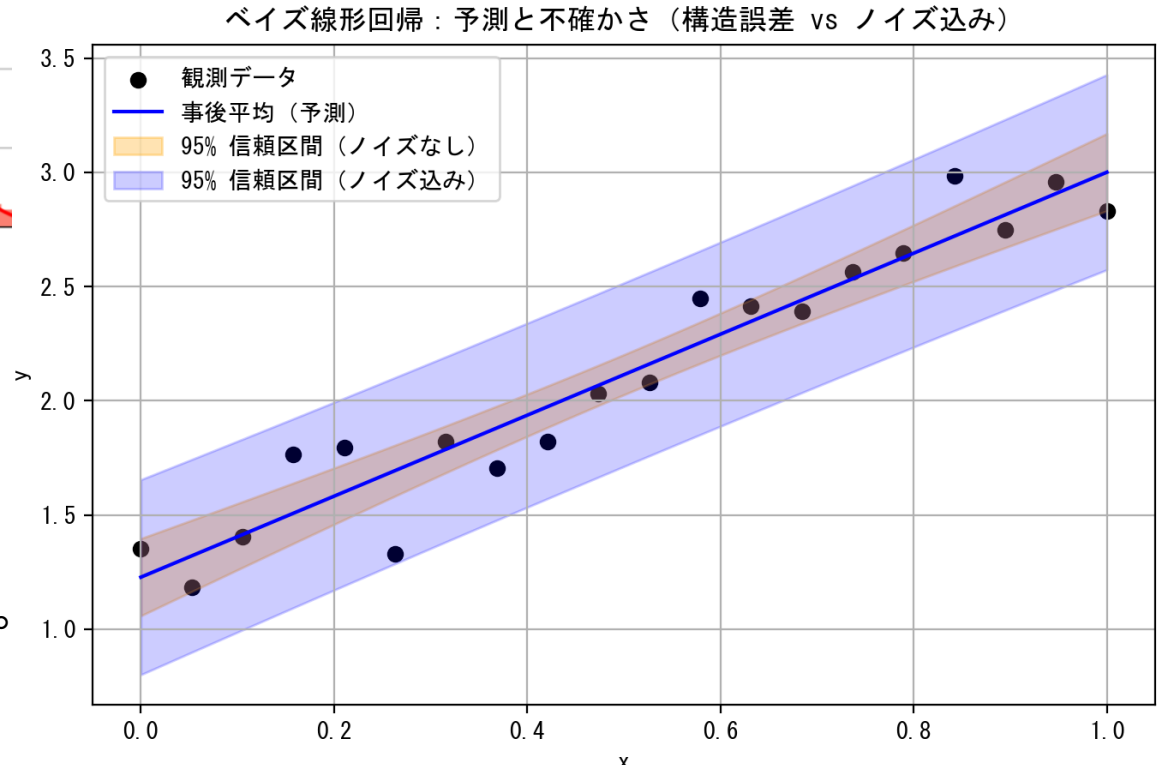
ベイズ線形回帰: プログラム

bayesian_linear_regression_mcmc.py

モデル関数: $y = a + bx$







ノイズなし誤差: 潜在関数（真の回帰関数）の不確かさ。
　パラメータの事後分布に基づく予測であり、観測ノイズは含まない。

ノイズ込み誤差: 観測値の予測分布の不確かさ。
　潜在関数の不確かさに加えて、観測ノイズのばらつきを含む。

ベイズ線形回帰・ベイズ更新: プログラム

bayesian_linear_regression_mcmc_animation.py

モデル関数: $y = a0 + a1 * x + a2 * x^2 + a3 * \mathrm{sqrt}(x)$
データ数を増やしながら、回帰結果とパラメータの分布をアニメーション表示