シミュレーション工学

モンテカルロ法(1) 基礎的な話題

慶應義塾大学大学院理工学研究科基礎理工学専攻物理情報専修

渡辺宙志

はじめに

モンテカルロ法とは

乱数を使った数値計算手法の総称
多くの場合「マルコフ連鎖モンテカルロ法」のこと
実装は比較的容易だが、**原理の理解は難しい**

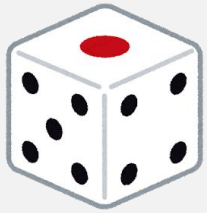
本講義の目的

モンテカルロ法の用語の意味を理解する
→ **特に「重み」について学ぶ**
マルコフ連鎖モンテカルロ法の手続きについて理解する
→「**なぜマルコフ連鎖モンテカルロ法が必要か**」を理解する

乱数とは

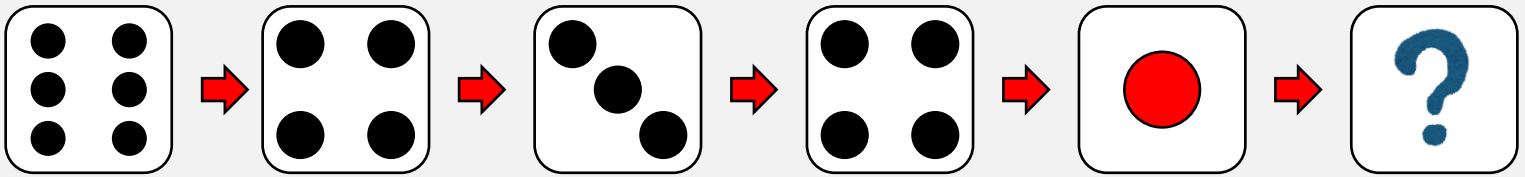
直感的な理解

ランダムな数のこと
例えばサイコロを振った時に出る目は乱数



真面目な定義

過去の数列$x_1, x_2, \cdots, x_n$から、次の数x_{n+1}が予想できない数列を**乱数列**と呼び、その要素を**乱数**と呼ぶ

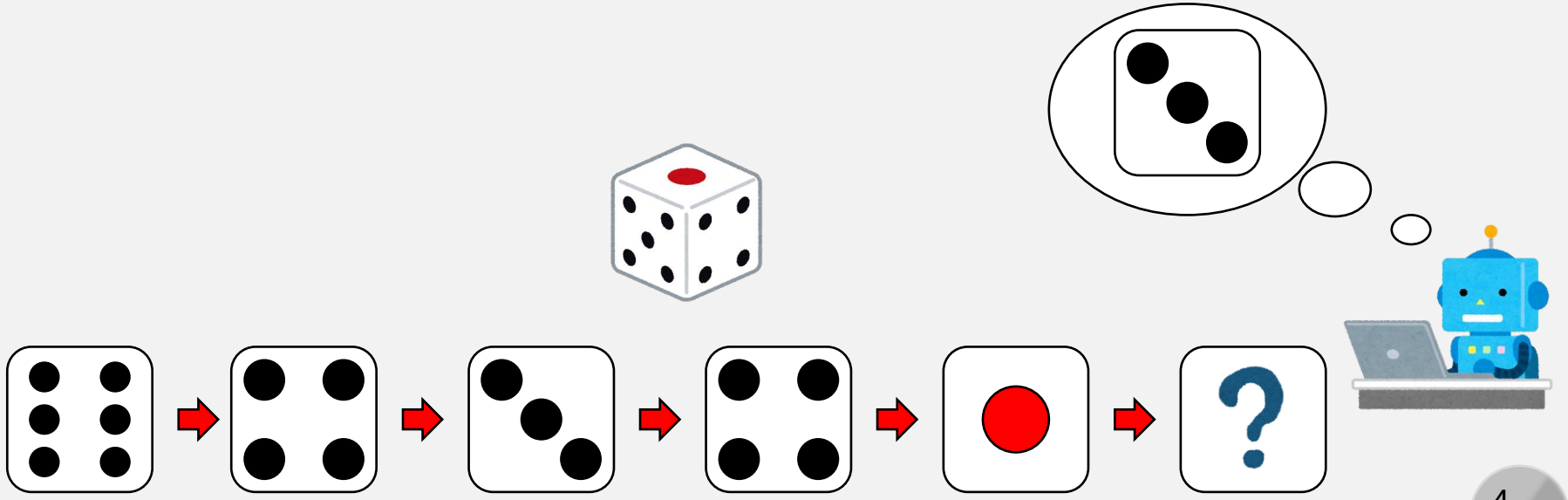


疑似乱数とは

疑似乱数(Pseudo Random Number)とは、履歴から**決定論的**に次の数字が決められている乱数

疑似乱数を生成するアルゴリズムと履歴がわかれば、原理的には「次の数」を**予想可能**

計算機で用いられる乱数は、ほぼ**疑似乱数**



疑似乱数生成アルゴリズム

線形合同法

$$X_{n+1} = (aX_n + b) \mod M$$

「一つ前」しか見ない
簡単・高速だが、乱数の性質は悪い

メルセンヌ・ツイスタ法

$$\vec{x}_{n+p} = \vec{x}_{n+q} + \vec{x}_{n+1}B + \vec{x}_n C$$

乱数の性質が非常に良い
多くの乱数ライブラリのデファクト・スタンダード

Xorshift法

```
x ^= (x << 13 & 0xFFFFFFFF)
x ^= (x >> 17 & 0xFFFFFFFF)
x ^= (x << 5 & 0xFFFFFFFF)
```

乱数の性質が比較的良い
非常に高速

疑似乱数と真乱数

真乱数

- 過去の履歴から次の数が**原理的に予測不可能**
- 熱雑音や放射性物質の崩壊など、物理現象を利用して真乱数を生成する装置を**物理乱数生成器**という

※ サイコロも物理乱数生成器の一種

疑似乱数

- **原理的に予測可能**
  - 十分な履歴があれば、次の数が予想可能
  - 同じ乱数の種から必ず同じ乱数列を得る
- 数値計算では以下の性質が重要
  - 十分に周期が長い(無相関性)
  - 出現する数に偏りがない(一様性)
- 数値計算では予測可能性は重視されない

乱数生成プログラムの例

```
import random
for _ in range(5):
    print(random.random())
```

```
$ python3 rand.py
0.7183142085184294
0.625356371754038
0.8206028825940407
0.5122008096362916
0.7253633754087734

$ python3 rand.py
0.3618195051209263
0.7496549080606681
0.3396919019733251
0.972292864599307
0.3532634875426808
```

実行するたびに異なる乱数列を得る

Pythonでは、種を指定しないと
種として「現在時刻」が用いられる

乱数生成プログラムの例

```
import random
random.seed(1)
for _ in range(5):
    print(random.random())
```

乱数の「種」を指定

```
$ python3 rand.py
0.13436424411240122
0.8474337369372327
0.763774618976614
0.2550690257394217
0.4954350870919409 5

$ pyton3 rand.py
0.13436424411240122
0.8474337369372327
0.763774618976614
0.2550690257394217
0.4954350870919409 5
```

毎回同じ乱数列が得られる

乱数生成プログラムの例

```cpp
#include <iostream>
#include <random>

int main() {
  std::mt19937 mt;
  std::uniform_real_distribution<> ud(0.0, 1.0);
  for (int i = 0; i < 5; i++) {
    std::cout << ud(mt) << std::endl;
  }
}
```

```
$ g++ rand.cpp
$ ./a.out
0.135477
0.835009
0.968868
0.221034
0.308167
```

← 毎回同じ結果を得る

C++では、種を指定しないと
デフォルトの「種」が指定される

乱数のまとめ

真乱数と疑似乱数

- 乱数列とは、履歴から次の数が予測不可能な数列
- 疑似乱数とは、履歴から次の数が生成されている数列
  →本質的には予測可能

疑似乱数と乱数の種

- 疑似乱数は同じ履歴から同じ数列を得る
- 乱数には「種」を与えることができる
- 同じ「種」から同じ乱数列を得る
- 分布が一様であり、相関が十分に小さければ、
  予測可能性は重視されない

※ むしろ、同じ種から同じ乱数列を得るのはデバッグで重要

モンテカルロ法

モンテカルロ法(Monte Carlo Method)とは
乱数を用いるシミュレーション手法

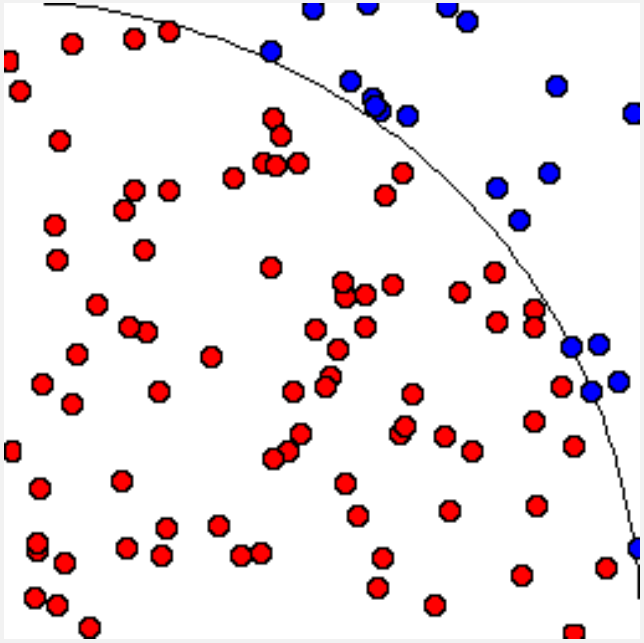
※ カジノで有名なモナコのモンテカルロに由来する

数値計算では、期待値や積分を近似的に求めるのに使われることが多い

モンテカルロ法の例：円周率の計算

円周率を求めるプログラム

```
import random
trial = 100000
n = 0
for _ in range(trial):
  x = random.random()
  y = random.random()
    if x**2 + y**2 < 1.0:
      n += 1
print(n/trial*4.0)
```



一辺1の正方形の領域に点をランダムにばらまく

$x^2 + y^2 < 1$ **を満たす確率は** $\pi/4$

モンテカルロ法の例：円周率の計算

先ほどのコードは以下の積分を実行していることに対応

$$\int_0^1 \int_0^1 \Theta(1 - x^2 - y^2)\, dx\, dy = \frac{\pi}{4} \qquad \Theta(x) = \begin{cases} 1 & x > 0 \\ 0 & x < 0 \end{cases}$$

- モンテカルロ法は**「式は書けるが厳密な評価は難しい和や積分」**のサンプリング評価に使われる
- 和や積分に表れる引数を一様にサンプリングすることを**単純サンプリング**と呼ぶ

単純サンプリング

これから「当たり前」の議論が延々続きますが、
後で必要になるのでちゃんとついてきてください

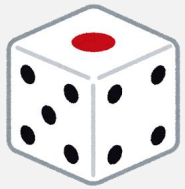
重み

公平なサイコロの目の期待値は？

状態kの値をY_kとする
状態kが出現する確率をp_kとする
状態は全部で6個

$$Y_1 = 1, Y_2 = 2, Y_3 = 3, Y_4 = 4, Y_5 = 5, Y_6 = 6$$

期待値を$\bar{X}$とすると



$$\bar{X} = \sum_{k=1}^{6} Y_k p_k = \sum_{k=1}^{6} k p_k$$

公平なサイコロなら$p_k = 1/6$だから

$$\bar{X} = \sum_{k=1}^{6} \frac{k}{6} = 3.5$$

重み

もしp_kを事前に知らなかったら？→ 多数回の試行によりp_kを推定する

サイコロをN回振って、kが出た回数をw_kとする

$$N = \sum_k w_k$$

kが出る確率p_kはw_kに比例するので

$$p_k \sim \frac{w_k}{N}$$

サイコロの目の期待値は

$$\bar{X} = \sum_{k=1}^{6} k p_k \sim \sum_{k=1}^{6} k \frac{w_k}{N}$$

w_kを状態kの重みと呼ぶ

重み

サイコロを何度も振り、i番目に出た目を$\hat{X}_i$とする

kが出た回数　$w_k = \sum_{i=1}^{N} \delta_{\hat{X}_i,k}$

クロネッカーのデルタ

$$\delta_{x,y} = \begin{cases} 1\ (x = y) \\ 0\ (x \neq y) \end{cases}$$

代入

$$\bar{X} = \sum_{k=1}^{6} k p_k \sim \sum_{k=1}^{6} k \frac{w_k}{N} = \frac{1}{N} \sum_{k=1}^{6} k \sum_{i}^{N} \delta_{\hat{X}_i,k}$$

$$k\delta_{\hat{X}_i,k} = \begin{cases} k\ (\hat{X}_k = k) \\ 0\ (\hat{X}_k \neq k) \end{cases}$$ であるから和を入れ替えてkについて先に和をとると

$$\bar{X} \sim \frac{1}{N} \sum_{i}^{N} \hat{X}_i$$ ←サイコロを何度も振って算術平均をとる

単純サンプリング

数値計算では**重みw_kは既知**だが、**確率p_kが未知**である
ことが多いが、その状態で期待値を推定したい

期待値を推定したい　$\bar{X} = \sum_{k=1}^{6} k p_k$

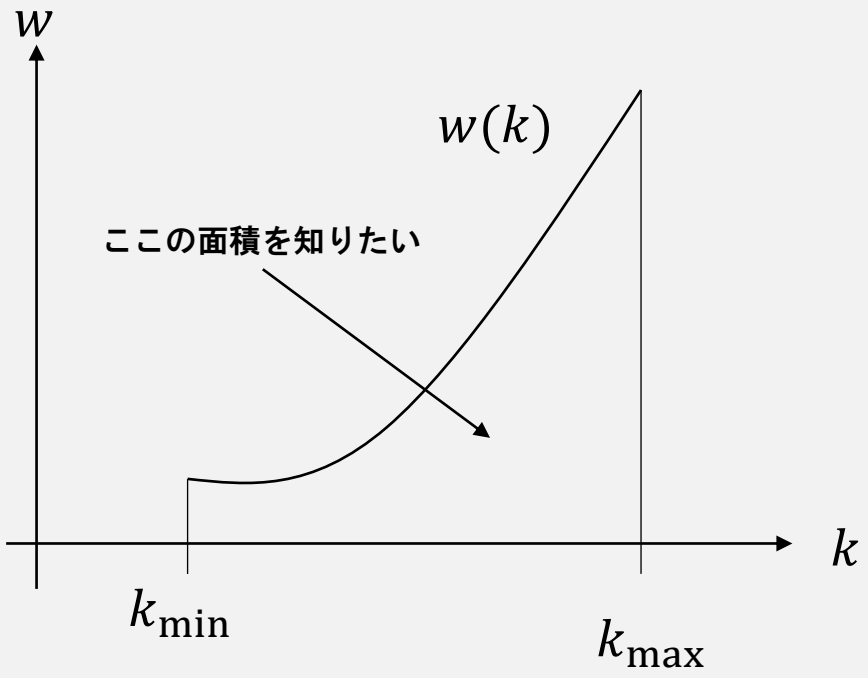
重みの総和をZとすると　$Z = \sum_{k} w_k$ ← これが計算できないから

確率は　$p_k = \frac{w_k}{Z}$ ← これがわからない

単純サンプリング

$$Z = \sum_k w_k$$

重みの総和が厳密に計算できない
→ サンプリングにより評価する



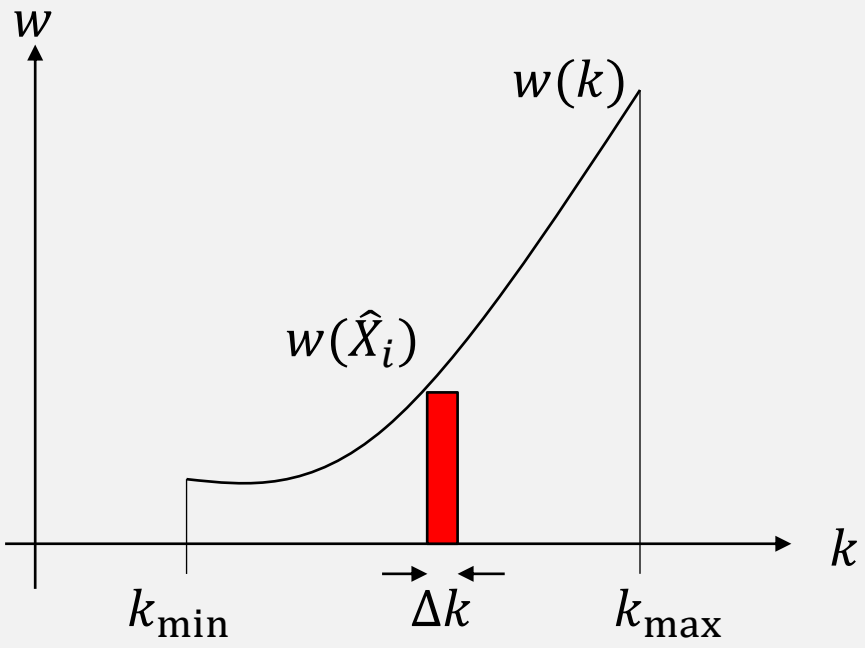
※和のままでも議論できるが、積分の方がわかりやすいのでそちらで

単純サンプリング

$k_{\min} < \hat{X}_i < k_{\max}$ を満たす一様乱数をN個生成

$\Delta k = \frac{k_{\max} - k_{\min}}{N}$ 短冊の幅

$$Z = \sum_k w_k \sim \sum_i w(\hat{X}_i)\Delta \mathrm{k}$$



単純サンプリング

同様に $\sum_k kw_k \sim \sum_i \hat{X}_i w(\hat{X}_i)\Delta \mathrm{k}$

以上から $\bar{X} = \sum_{k=1}^{6} kp_k = \frac{\sum_k kw_k}{\sum_k w_k} \sim \frac{\sum_i \hat{X}_i w(\hat{X}_i)}{\sum_i w(\hat{X}_i)}$

以上のように、重みに関係なく状態候補をランダムに選び、
選んだあとに重みをかけて平均をとる手法を**単純サンプリング**と呼ぶ

サイコロの場合、重みが等しいので$w_k = 1$とすると

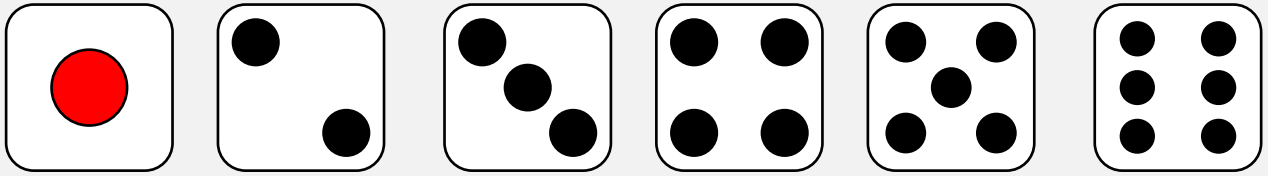
$$\bar{X} = \frac{\sum_i \hat{X}_i w(\hat{X}_i)}{\sum_i w(\hat{X}_i)} \sim \frac{1}{N}\sum_{i=1}^{N} \hat{X}_i$$

単純サンプリングのまとめ

- 状態の重みとは出現確率に比例するもの
- 状態の重みは既知だが、重みの総和が求まらないため、状態の出現確率が未知であることが多い
- 何かを調べたいとき、すべてを調べるのではなく、一部の標本を抜き出して調べることをサンプリングと呼ぶ
- 何かの期待値を乱数を使って評価する手法をモンテカルロ法と呼ぶ
- 出現可能な状態から(重みと無関係に)一様に状態を選び、重みをかけてから期待値を推定する方法を単純サンプリングと呼ぶ
- 単純サンプリングはモンテカルロ法の最も簡単な例

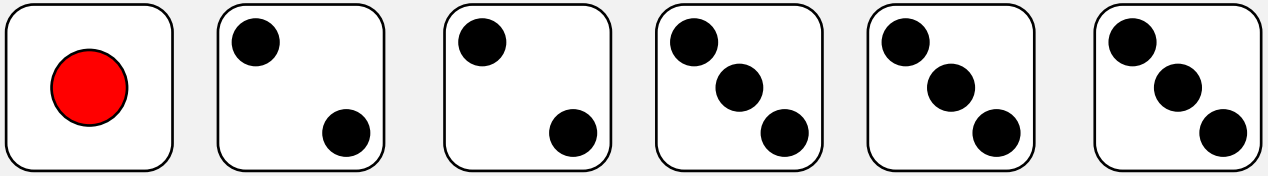
重複のあるサイコロ

普通のサイコロ



$$Y_1 = 1, Y_2 = 2, Y_3 = 3, Y_4 = 4, Y_5 = 5, Y_6 = 6$$

重複サイコロ



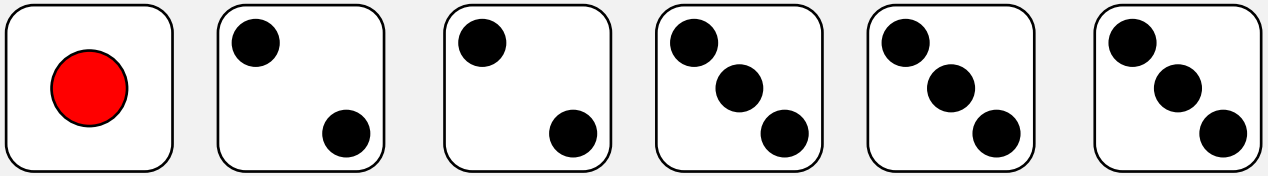
$$Y_1 = 1, Y_2 = 2, Y_3 = 2, Y_4 = 3, Y_5 = 3, Y_6 = 3$$

期待値

$$\bar{X} = Z^{-1} \sum_{k=1}^{6} Y_k w_k \qquad Z = \sum_{k} w_k$$

重複のあるサイコロ

重複サイコロ



$$Y_1 = 1, Y_2 = 2, Y_3 = 2, Y_4 = 3, Y_5 = 3, Y_6 = 3$$

状態は6つあるが、値は3種類しかない
→同じ値をとる状態をまとめる
値jをとる状態数をg_jとすると

$$\bar{X} = Z^{-1} \sum_{k=1}^{6} Y_k w_k$$

$$Z = \sum_{k=1}^{6} w_k$$

状態に関する和を
値に関する和に
とりなおす

$$\bar{X} = Z^{-1} \sum_{j=1}^{3} j g_j w_j$$

$$Z = \sum_{j=1}^{3} w_j$$

重複のあるサイコロ

重複サイコロは値に重複があるが、重みは全て等しい
→　$w_j' = g_j w_j$という新たな重みを考える
→　値により重みが異なる「不公平なサイコロ」になる

重複のある公平なサイコロ

重複のない
不公平なサイコロ

状態j	$j=1$	$j=2$	$j=3$	$j=4$	$j=5$	$j=6$	$j=1$	$j=2$	$j=3$
値Y_j	$Y_1=1$	$Y_2=2$	$Y_3=2$	$Y_4=3$	$Y_5=3$	$Y_6=3$	$Y_1=1$	$Y_2=2$	$Y_3=3$
重みw_j	$w_1=1$	$w_2=1$	$w_3=1$	$w_4=1$	$w_5=1$	$w_6=1$	$w'_1=1$	$w'_2=2$	$w'_3=3$

重複のあるサイコロ

重複のない不公平なサイコロの期待値

$$\bar{X} = Z^{-1} \sum_{k=1}^{3} k w'_k$$

c.f. **重複のある公平なサイコロの期待値**

$$\bar{X} = Z^{-1} \sum_{j=1}^{3} j g_j w_j$$

期待値を単純サンプリングで求める手続き

$$\bar{X} = \frac{\sum_k k w'_k}{\sum_k w'_k} \sim \frac{\sum_i \hat{X}_i w'(\hat{X}_i)}{\sum_i w'(\hat{X}_i)}$$

1. 1,2,3のいずれかの値を一様に取る確率変数$\hat{X}_i$をN個生成
2. 出現した値に対応する重み$w(\hat{X}_i)$をかけて和をとる(分子の推定)
3. 出現した値に対応する重みの和をとる(分母の推定)
4. 十分な和がとれたら、それらの比をとる(期待値の推定)

※ 式は公平なサイコロと同じだが、重みが一様でない例になっている

単純サンプリングのまとめ

用語の整理

- 値jを持つ状態の重みがw_j
- 値jを持つ状態の状態数がg_jである時、
- 重みの総和を$Z = \sum_j g_j w_j$として
- 値の期待値は$\bar{X} = Z^{-1} \sum_j g_j w_j$

単純サンプリングの手続き

$$\bar{X} = \frac{\sum_k k w'_k}{\sum_k w'_k} \sim \frac{\sum_i \hat{X}_i w'(\hat{X}_i)}{\sum_i w'(\hat{X}_i)}$$

1. 状態候補$\hat{X}_i$を無作為に(重みに無関係に一様に)多数生成
2. 出現した値に対応する重み$w(\hat{X}_i)$をかけて和をとる(分子の推定)
3. 出現した値に対応する重みの和をとる(分母の推定)
4. 十分な和がとれたら、それらの比をとる(期待値の推定)

マルコフ連鎖モンテカルロ法

Markov Chain Monte Carlo (MCMC) method

- 非常に効率が良い
- バイアスがない
- 数値計算における「モンテカルロ法」といえばこれ
- コードは簡単だが、原理の理解は難しい(※個人の感想)

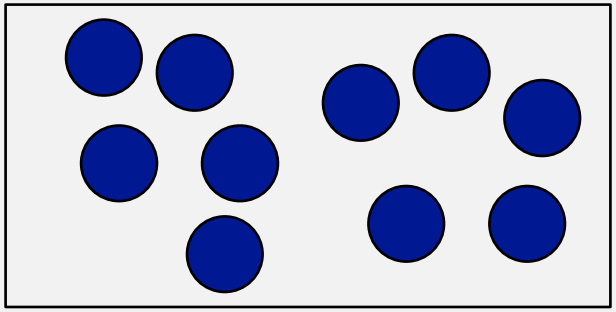
疑問点

- マルコフ連鎖モンテカルロ法とは何か？
- なぜマルコフ連鎖が必要なのか？
- 詳細釣り合い条件とは何か？
- そもそも何を計算しているのか？

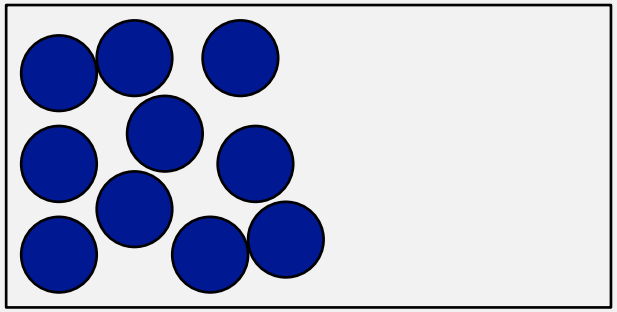
これらの疑問への回答を試みる

気液相転移

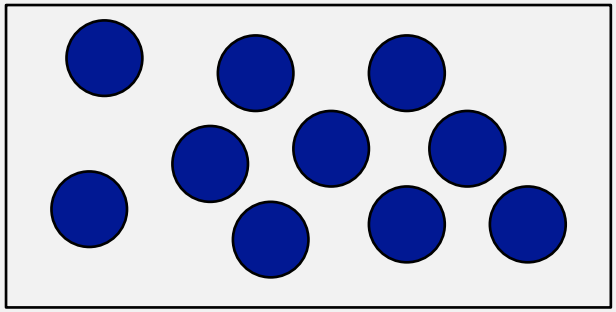
箱の中に原子を入れてしばらく放っておく



低温なら偏る
(エネルギー重視)



高温ならばらける
(エントロピー重視)

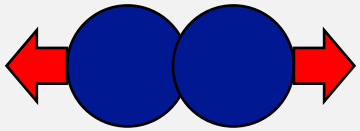


この相転移の様子を調べたい

気液相転移

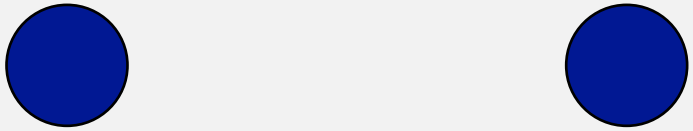
原子の相互作用をモデル化

近距離で斥力 (排除体積効果)



中近距離で引力 (ファンデルワールス力)

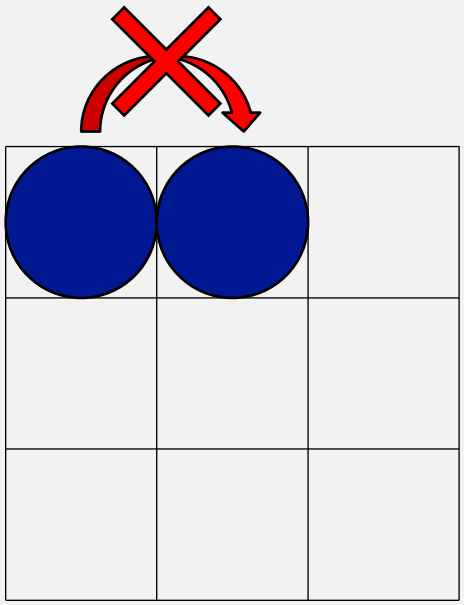
遠距離で相互作用なし



格子ガス模型

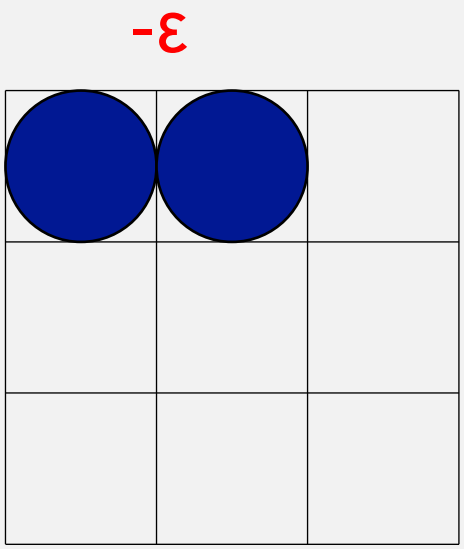
さらに空間を離散化したモデル

近距離で斥力



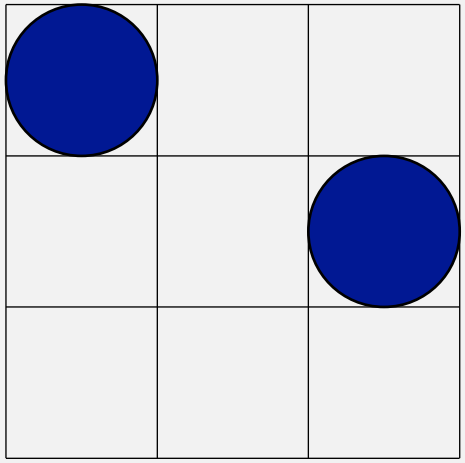
ひとつのセルに
２つの原子は入れない

中距離で引力



隣り合うとエネルギーが
εだけ下がる

遠距離で
相互作用無し



隣接していないと
相互作用なし

ボルツマン重み

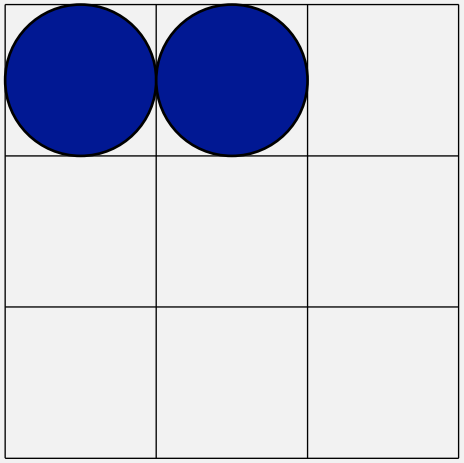
ある状態のエネルギーをE、温度をTとすると、
その状態の出現確率は以下に比例する

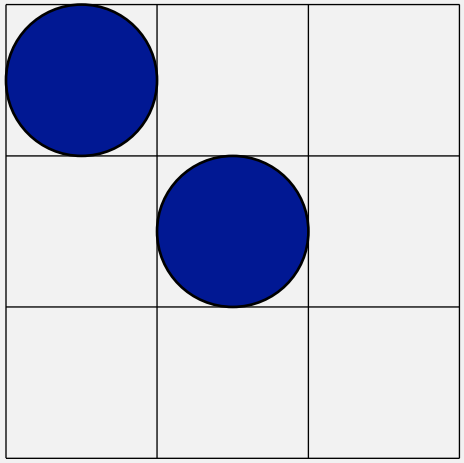
$$\exp(-\beta E)$$

ボルツマン定数 k_B

逆温度 $\beta = 1/k_B T$

-ε



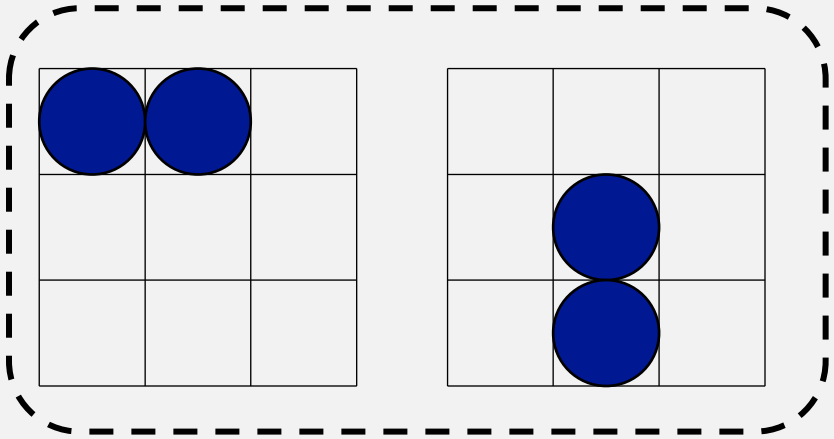


液相(左)が出現する確率は気相(右)が出現する確率の

$\exp(\beta\epsilon)$ 倍 ➡ 液相の出現確率の方が大きい

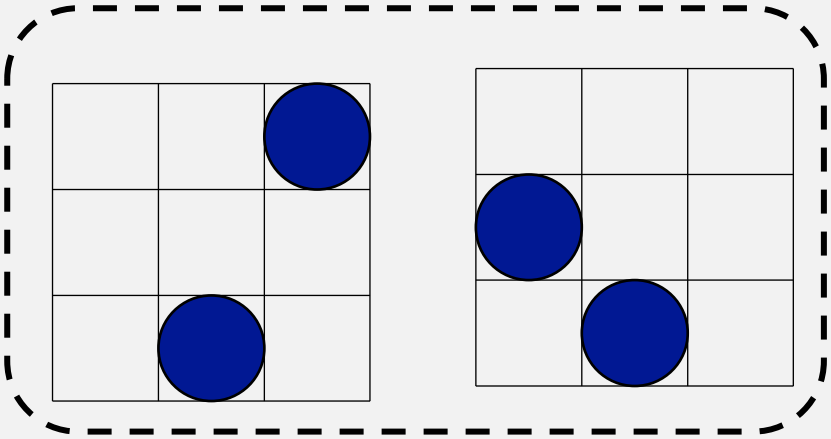
エネルギーの振る舞い

液相



一つ一つの出現確率は高いが、
総数が少ない
→エネルギー重視
→低温で支配的であろう

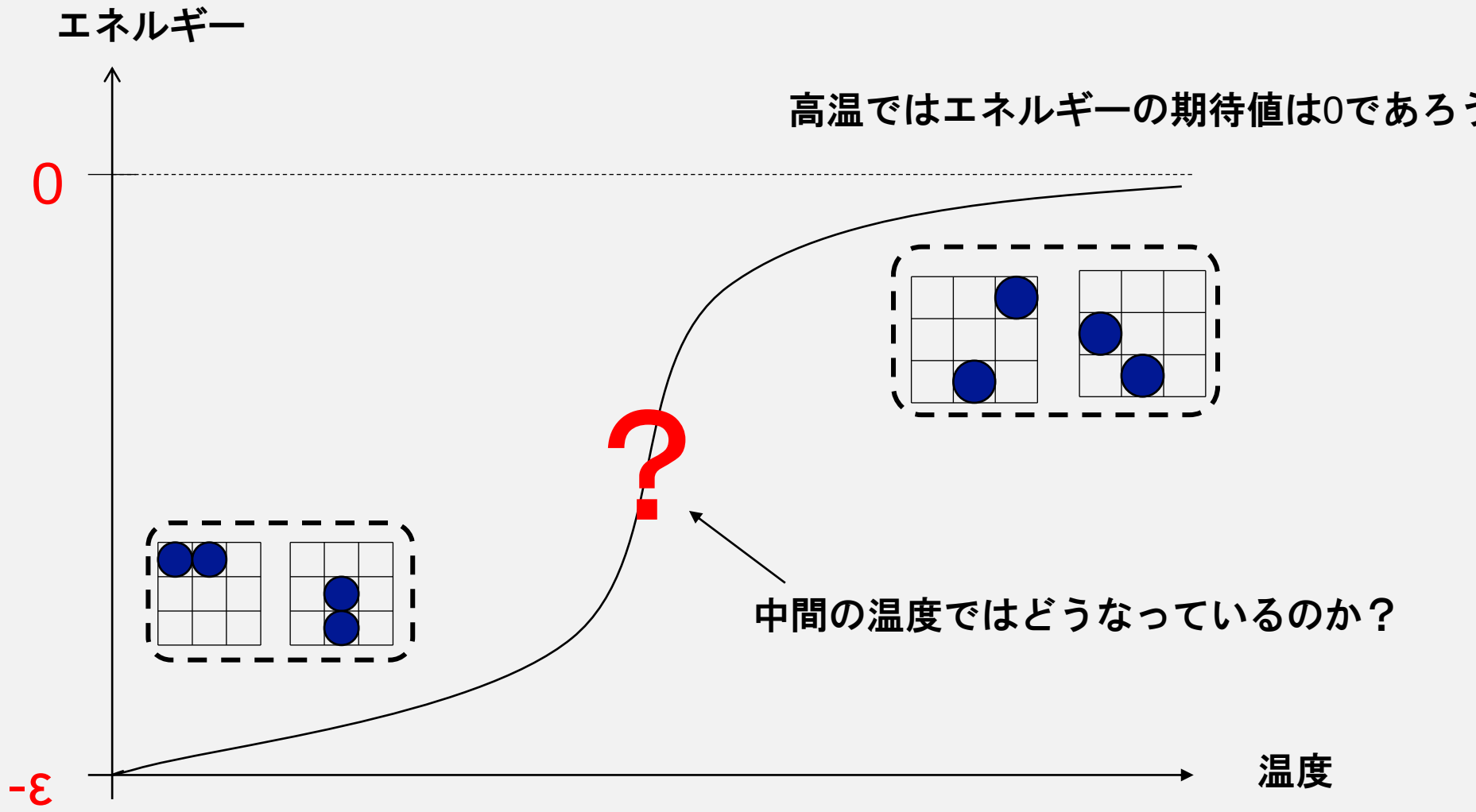
気相



一つ一つの出現確率は低いが、
総数が多い
→エントロピー重視
→高温で支配的であろう

エネルギーの温度依存性$U(T)$を知りたい
まずは2原子系で考える

エネルギーの振る舞い



低温ではエネルギーの期待値は-εであろう

エネルギーの振る舞い

E_i 状態iにおけるエネルギー

$w_i = \exp(-\frac{E_i}{kT})$ 状態iが出現する重み

$p_i = \frac{w_i}{\sum_i w_i}$ 状態iをとる確率

$U(T) = \sum_i E_i p_i$ 温度Tにおけるエネルギーの期待値

これを様々な温度で計算したい

エネルギーの振る舞い

$$U(T) = \sum_i E_i p_i$$

「状態の和」になっていると扱いが難しい

- 同じエネルギーを持つ状態が多数ある
- 出現確率はエネルギーにのみ依存する

→ 同じエネルギーを持つ状態について和をまとめる

$$U(T) = \sum_E Eg(E)p(E)$$

「エネルギーに関する和」に取り直した

エネルギーの振る舞い

$$U(T) = \sum_E Eg(E)p(E)$$ エネルギーに関する和

$$g(E)$$ エネルギーEをとる状態の数(Density of State, DoE)

$$p(E) = \frac{W(E)}{Z}$$ エネルギーがEである(ひとつの)状態が出現する確率

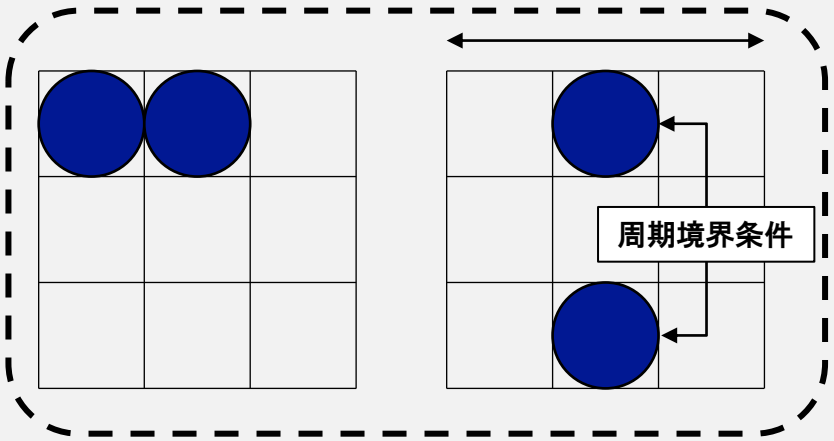
$$W(E) = \exp\left(-\frac{E}{kT}\right)$$ ボルツマン重み

$$Z = \sum_i w_i = \sum_E g(E)W(E)$$ 全ての重みの和(**分配関数**)

状態数を計算してみる

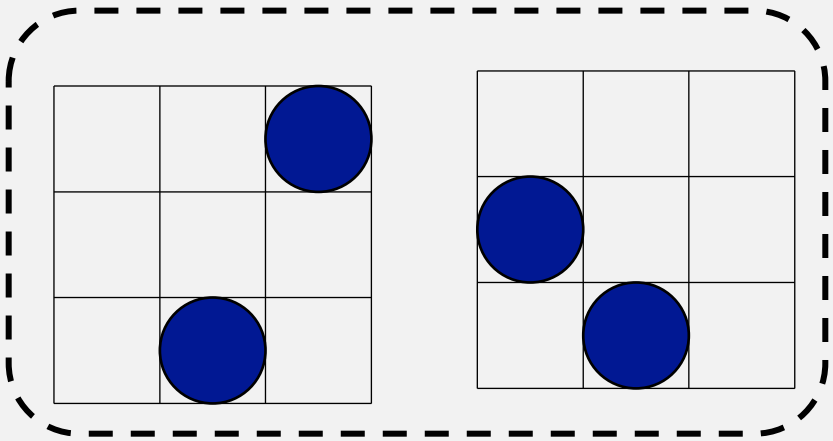
V=L x Lの格子を考える

液相



$2V$ 通り

気相



$\frac{V(V-1)}{2} - 2V$ 通り

$$g(-\epsilon) = 2V$$

$$g(0) = \frac{V(V-1)}{2} - 2V$$

状態数を計算してみる

$$U(T) = \sum_E Eg(E)p(E)$$

$$= \frac{-\epsilon g(-\epsilon)w(-\epsilon) + 0g(0)w(0)}{g(-\epsilon)w(-\epsilon) + g(0)w(0)}$$

分配関数

2原子が隣接する状態数

$$g(-\epsilon) = 2V$$

それ以外の状態数

$$g(0) = \frac{V(V-1)}{2} - 2V$$

2原子が隣接する状態の重み

$$w(-\epsilon) = \exp\left(\frac{\epsilon}{kT}\right)$$

それ以外の重み

$$w(0) = 1$$

必要なものが全てそろったので厳密に計算できる

状態数を計算してみる

$$w(-\epsilon) = \exp\left(\frac{\epsilon}{kT}\right)$$

温度はここにしか出てこない
エネルギーと温度は必ずセットで出てくる

$$K = \frac{\epsilon}{kT}$$

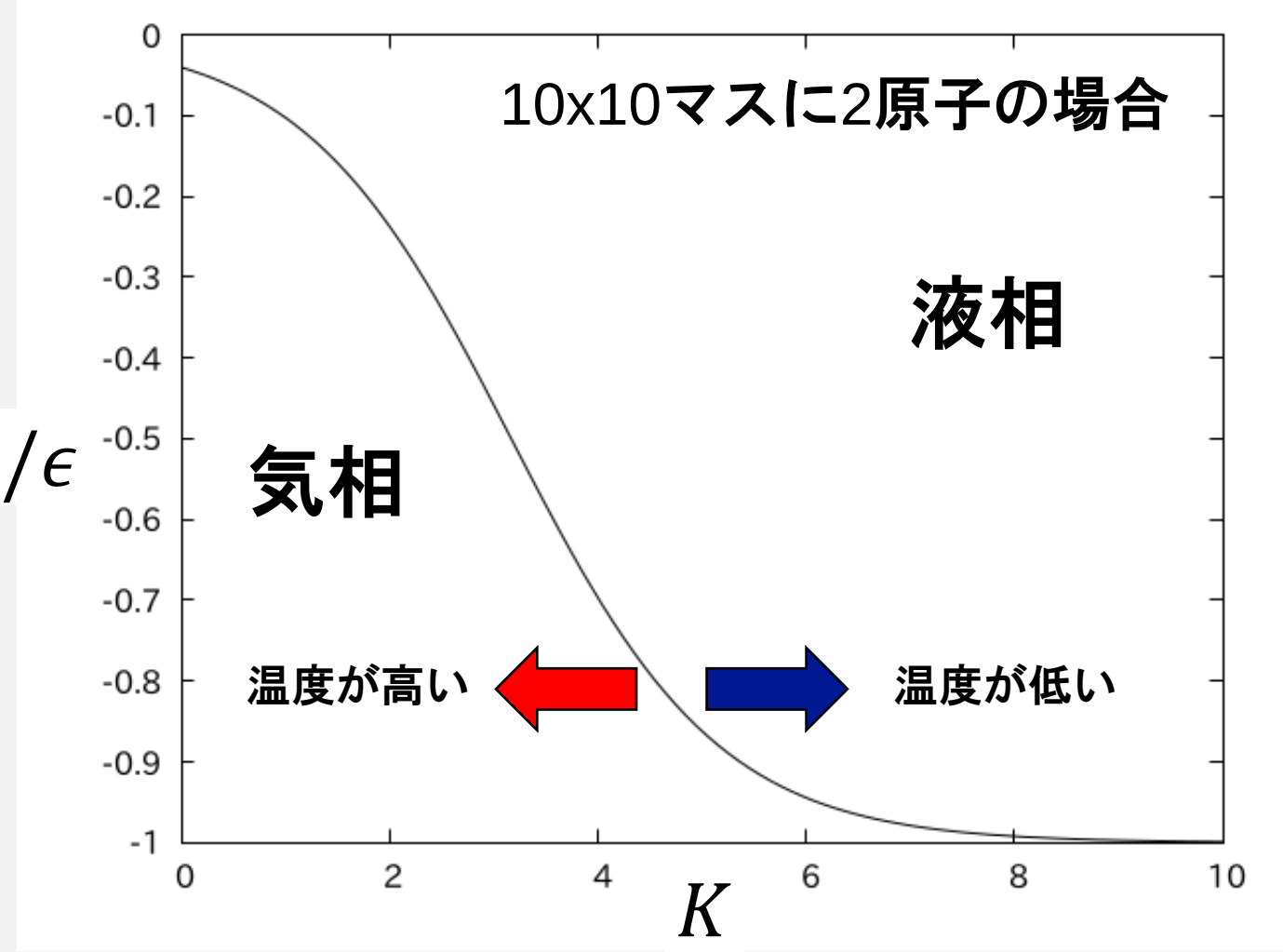
無次元化された逆温度を導入する

高温、弱相互作用→Kが小きい
低温、強相互作用→Kが大さい

$$U(K) = \frac{-\epsilon V e^K}{-\epsilon V e^K + V(V-5)/2}$$

エネルギーの(逆)温度依存性が厳密に求まった

重み vs. 状態数



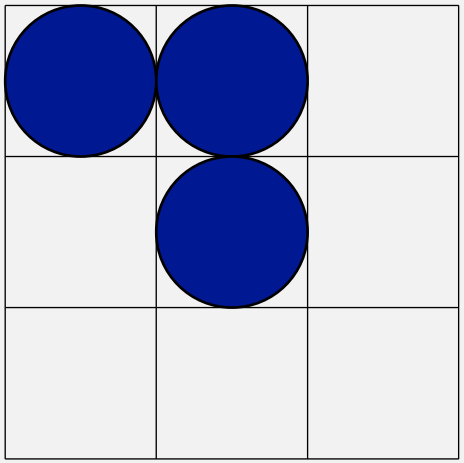
$$K = \frac{\epsilon}{kT}$$

サイズが大きくなるほど、原子が増えるほど、
「確率の入れ替わり」が急峻に→相転移

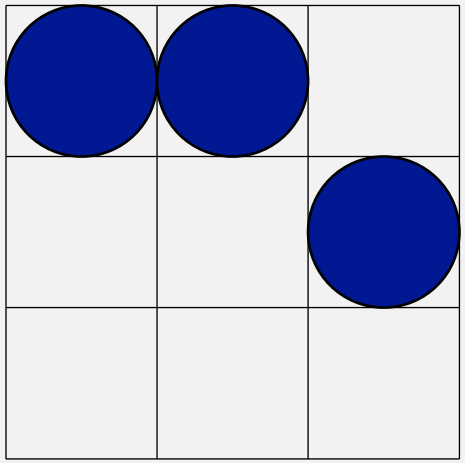
原子が増えると？

3原子の場合、取り得るエネルギーは3種類になる

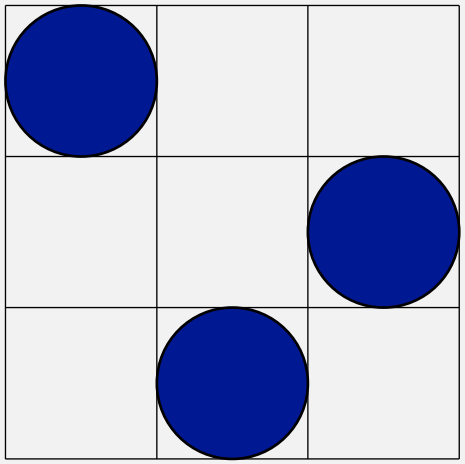
-2ε



-ε



0



3原子ならなんとかなるが、一般のN原子系では絶望的

単純サンプリング

$$p_i = \frac{w_i}{\sum_i w_i}$$

確率を知るには、重みの総和が必要

$$\sum_i w_i = \sum_E g(E) W(E) \equiv Z$$

重みの総和(分配関数)を求めるには
状態数$g(E)$が必要

- 状態jからエネルギーE_jや重みw_jは計算できる
- エネルギーEから、そのエネルギーを持つ状態数$g(E)$はわからない
- 状態数がわからないので、状態iが出現する確率p_iもわからない
- 状態の出現確率p_iがわからない状態で、以下の量を推定したい

$$U(T) = \sum_i E_i \, p_i$$

➡ サンプリングによる推定

単純サンプリング

1. V個のサイトから無作為にN個選び、そこに原子を置いた状態をiとする
2. 状態iのエネルギーE_iを計算する
3. エネルギーから重みw_iを計算する
4. 以上を繰り返し$\sum_i E_i w_i$と$\sum_i w_i$の比を計算する

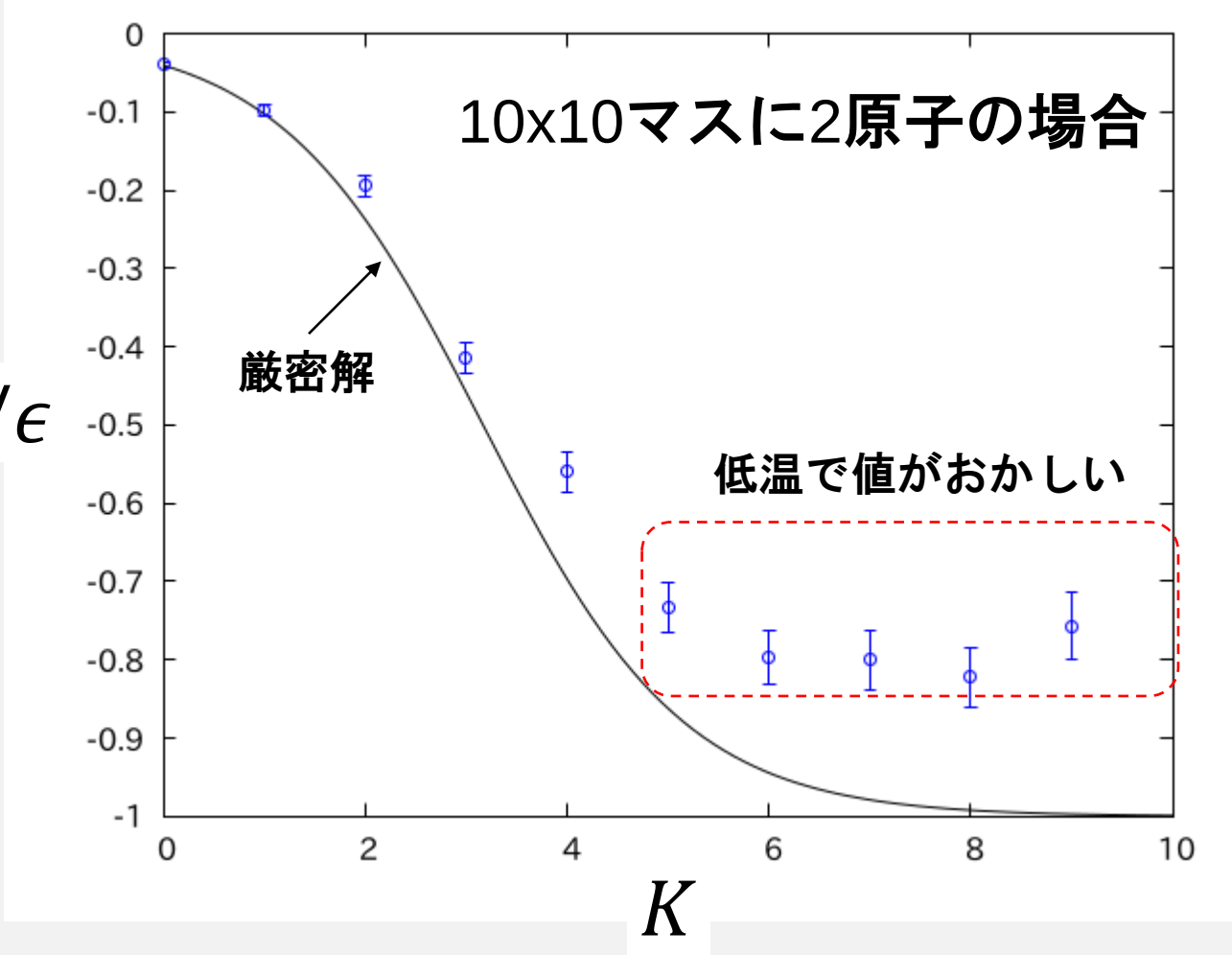
$$U(T) = \sum_k E_k p_k \sim \frac{\sum_i E_i w_i}{\sum_i w_i}$$

全ての状態についての和
(厳密)

ランダムに生成した状態についての和
(サンプリング)

単純サンプリング

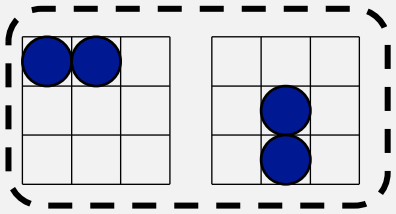
単純サンプリングの数値計算例
各温度で状態を50回生成する手続きを100サンプル平均



単純サンプリング

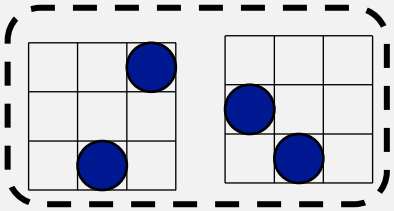
何が起きた？

液相



状態数が少ないが
重みが大きい

気相



状態数が多いが
重みが小さい

- 状態をランダムに生成すると、エネルギーが低い状態が選ばれる確率が極めて低くなる
- 温度が低い場合、ほとんどの寄与はエネルギーが低い状態からくる
- 極めて低確率で出現する状態が、極めて大きな重みを持つ
  → 収束に非常に多数の試行数が必要となる

例：宝くじの期待値

マルコフ連鎖モンテカルロ法

単純サンプリングは、重みを無視して状態を生成していたのが問題
→重みに比例して状態を生成したい

もし状態iを、重みw_iに比例して生成できたら？

$$p_i = \frac{w_i}{\sum_i w_i}$$

状態jの出現確率

$$U(T) \sim \frac{1}{M}\sum_{j}^{M} E_j$$

M回状態を生成して
エネルギーを平均するだけ

しかし、重みの総和は計算できない

重みの総和を計算しないまま、重みに比例して状態を生成したい

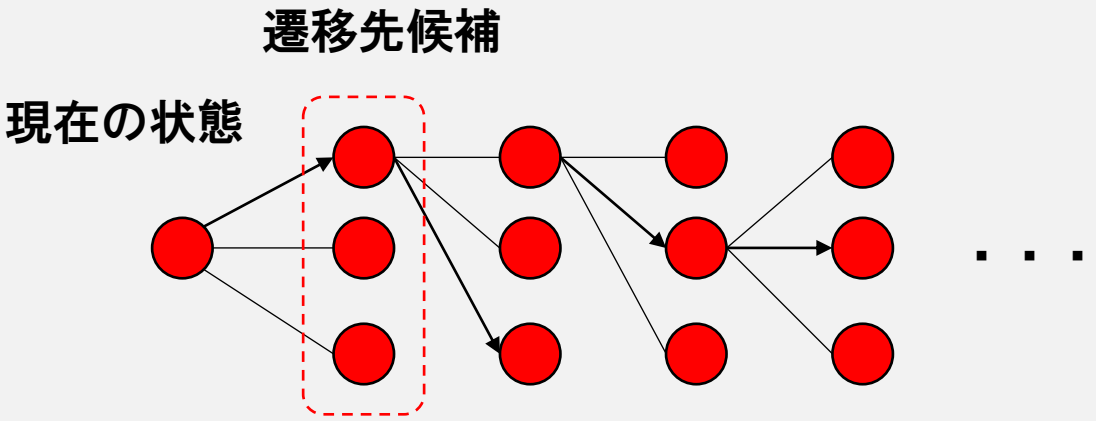
➡ マルコフ連鎖モンテカルロ法

マルコフ連鎖モンテカルロ法

- 単純サンプリングでは全ての状態候補の中から無作為に選んでいた
- 重みの高い状態を優先的に選びたい
- 重みの総和がわからないため、「全ての状態候補」から標本を選ぶことはできない

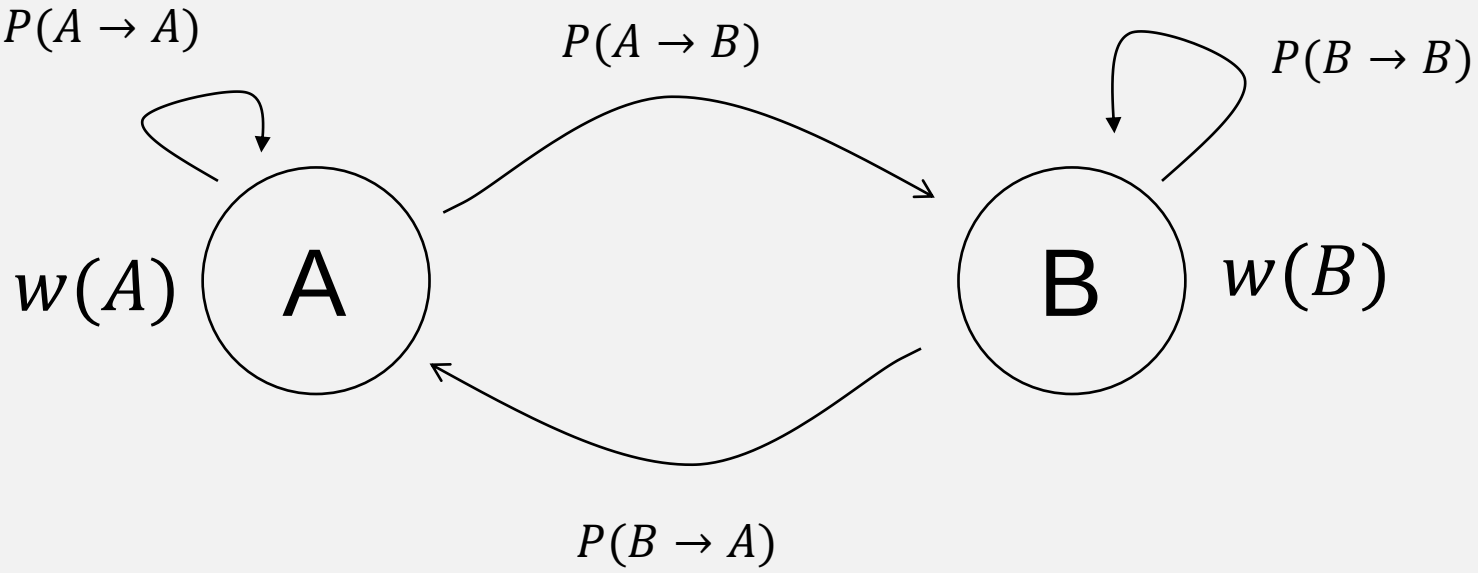
➡ 選ぶ状態候補を限定しよう

- 「現在の状態」から遷移できる状態を限定する
- その中から「遷移先候補」を選ぶ
- 「現在の状態」と「遷移先候補」の重みから、遷移確率を計算する
- 遷移確率から遷移させるかどうか決める
- こうして次々と状態を連鎖的に生成する



マルコフ連鎖モンテカルロ法

二つの状態A,Bを考える
それぞれ重み$w(A), w(B)$ を持っている
平衡状態における分布$\pi(A)$と$\pi(B)$が重みに比例するように遷移確率を決めたい



Aの人口　　AからBに行く割合

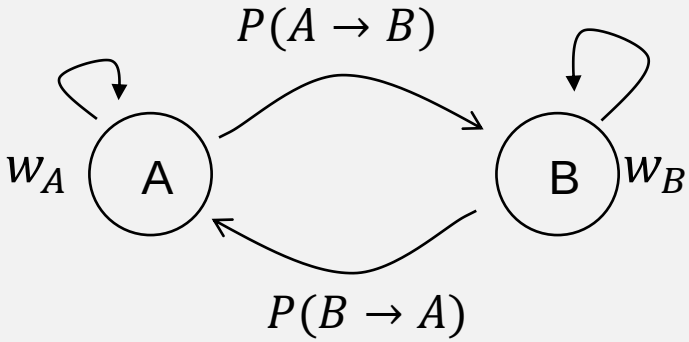
平衡状態では　$\pi(A)P(A \to B) = \pi(B)P(B \to A)$

AからBに行く人口　　BからAに行く人口

マルコフ連鎖モンテカルロ法

目的

$\pi(A) \propto w(A), \pi(B) \propto w(B)$ となるように
$P(A \to B), P(B \to A)$を決める



$$\pi(A)P(A \to B) = \pi(B)P(B \to A)$$

より

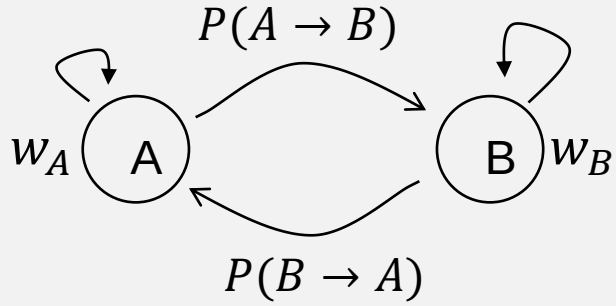
$$\frac{P(A \to B)}{P(B \to A)} = \frac{\pi(B)}{\pi(A)} = \frac{w(B)}{w(A)}$$

を満たすように決めればよい

この条件が全ての遷移可能な状態間で満たされることを
詳細釣り合い条件(Detailed Balance Condition)と呼ぶ

マルコフ連鎖モンテカルロ法

$$\frac{P(A \to B)}{P(B \to A)} = \frac{\pi(B)}{\pi(A)} = \frac{w(B)}{w(A)}$$



を満たす遷移確率の決め方は一意に決まらない
→適当に決める

熱浴法(heat-bath method)

素直に遷移確率も重みに比例させる

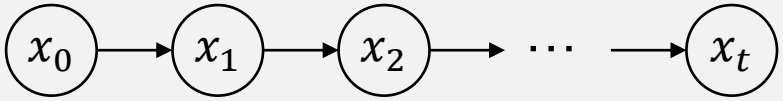
$$P(A \to B) = \frac{w(B)}{w(A) + w(B)}, P(B \to A) = \frac{w(A)}{w(A) + w(B)}$$

メトロポリス法(Metropolis method)

重みが大きい場合は必ず遷移、そうでない場合は確率的に遷移させる

$$P(A \to B) = 1, P(B \to A) = \frac{w(A)}{w(B)} \qquad w(A) < w(B)$$

マルコフ連鎖モンテカルロ法



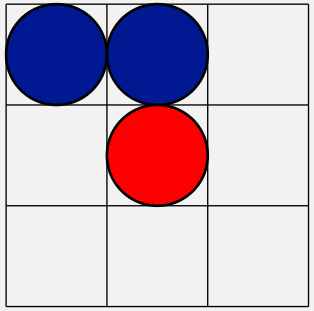
- 適当な状態x_0を決める
- そこから遷移可能な状態x'を適当に決める
- 重み$w(x_0)$と$w(x')$から、遷移確率を決める
- 遷移した場合は状態を更新、遷移しなかった場合は
- 現在の状態のままとし、それをx_1とする
- 同様にx_tからx_{t+1}を生成する(連鎖)

十分に緩和した場合、状態xの出現確率は重み$w(x)$に比例する(※)
状態xから遷移可能な状態yの間には、詳細釣り合いが満たされている
→ 状態yの出現確率も$w(y)$に比例する

※厳密に証明可能だが、今回は詳細に触れない

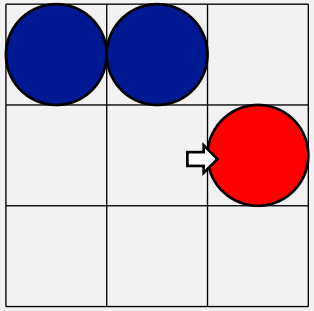
格子ガス系への適用

現在の状態x_t



適当な原子を一つ選ぶ

提案状態x'



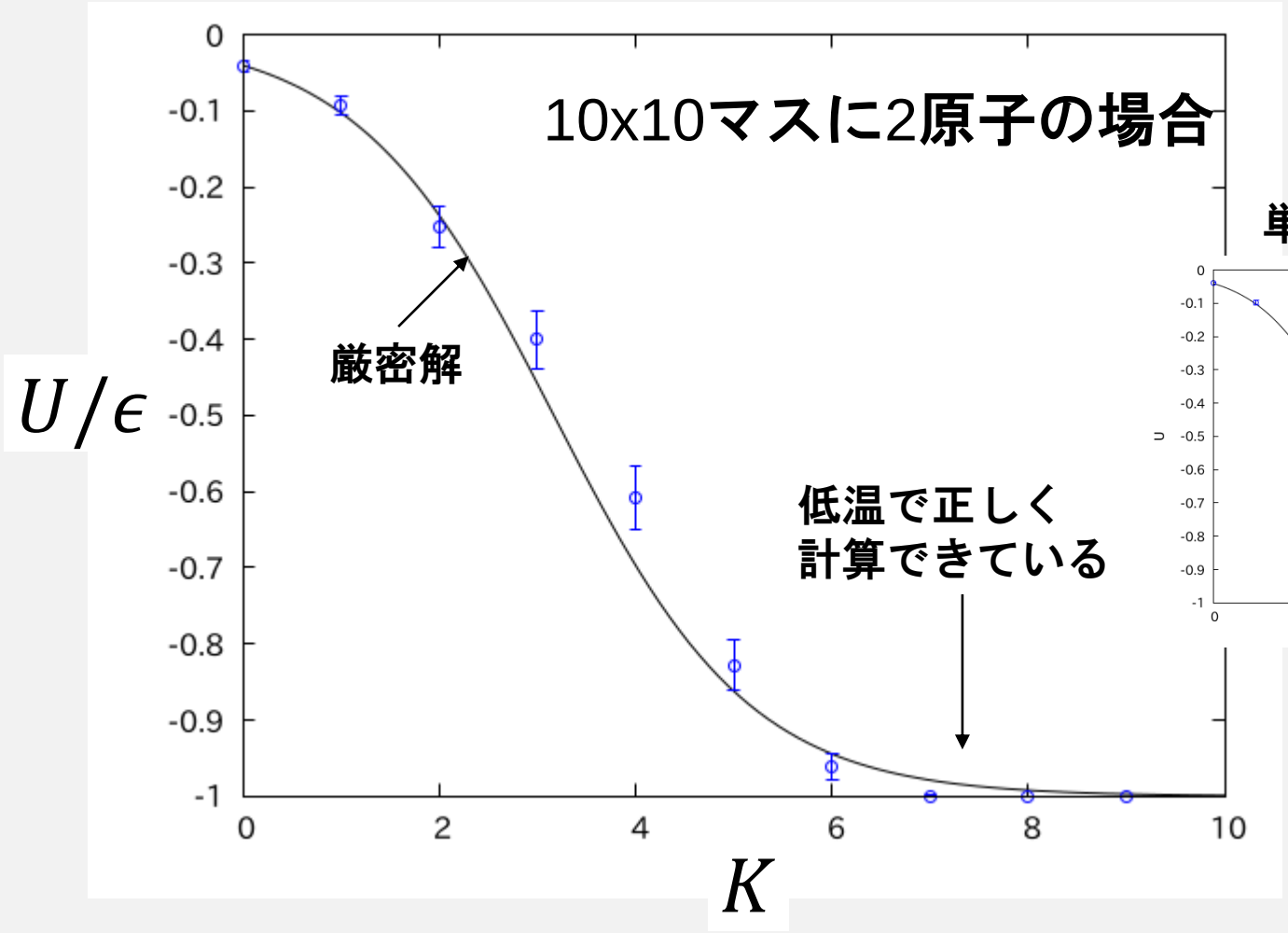
適当に動かす

エネルギーを計算し、エネルギーから重み$w(x')$を計算する
現在の重み$w(x_t)$と、提案状態の重み$w(x')$から、遷移させるかどうか決める
遷移してもしなくても、それを次の状態x_{t+1}とする

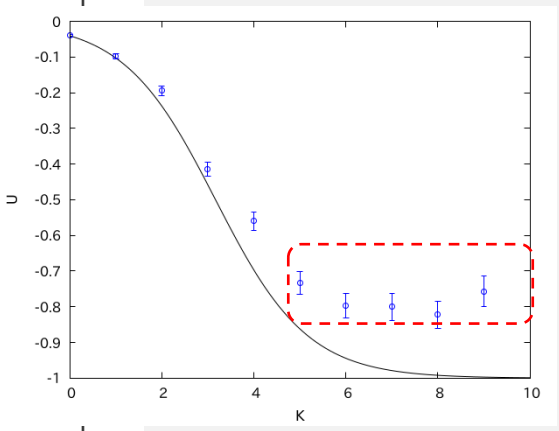
温度が低い場合は、エネルギーの高い場合に遷移しづらくなる
→エネルギーの低い状態にとどまり続ける
→エネルギーの低い状態が提案される確率が高くなる
→重みに比例して状態をサンプリングできる

格子ガス系への適用

マルコフ連鎖モンテカルロ法の数値計算例
各温度で状態を50回生成する手続きを100サンプル平均



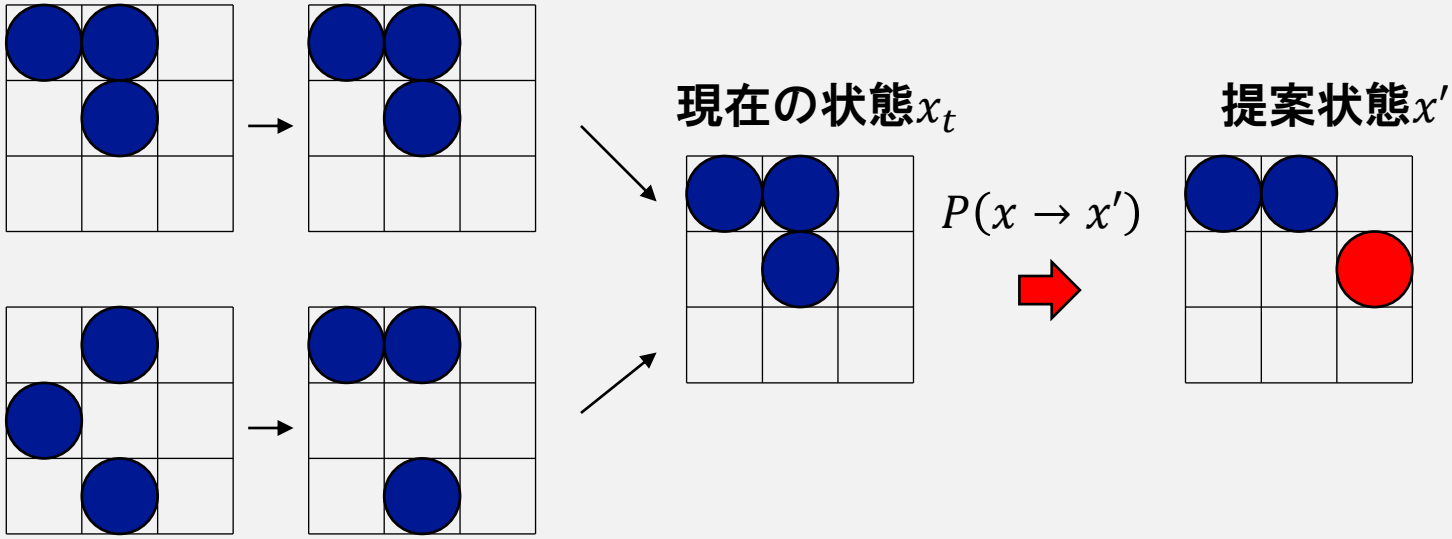
単純サンプリング



マルコフ性

遷移確率が現在の状態にのみ依存し、履歴に依存しないこと

t-2　　t-1　　t



過去にどのような
履歴をたどっていても

現在の状態が
同じなら

同じ提案状態への
遷移確率は等しい

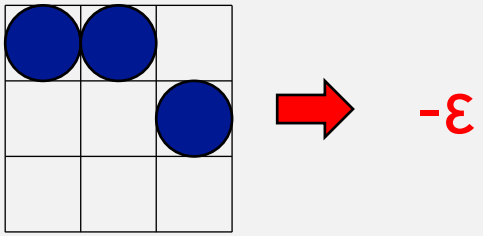
まとめ

マルコフ連鎖モンテカルロ法とは何か？

重みに比例するように提案状態を生成することで効率よくサンプリングする手法

なぜマルコフ連鎖モンテカルロ法が必要か？

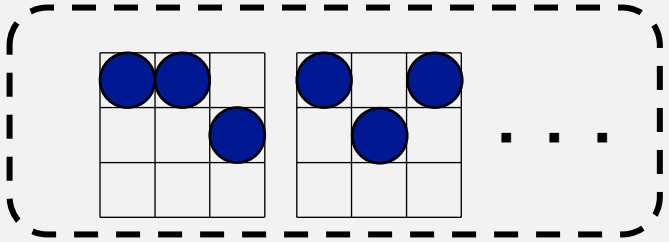
状態が与えられたらエネルギーの計算は容易だが、逆にエネルギーが与えられた時の状態数の計算が困難だから



-ε

状態からエネルギーを求めるのは簡単

-ε



エネルギーから状態数を求めるのは困難