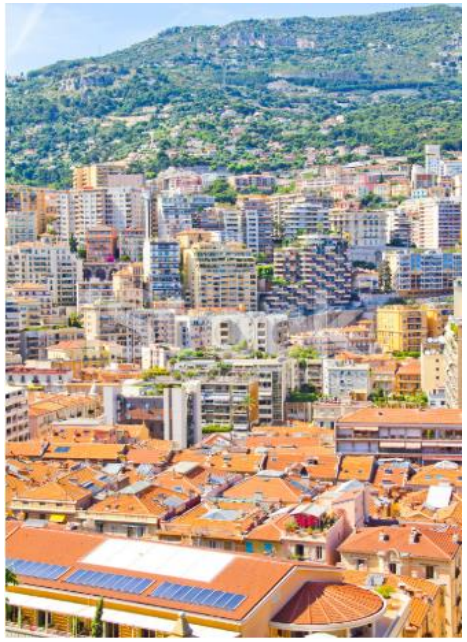
This week

Chapter 17

< Monte Carlo Methods >

1. Origin of Monte Carlo Method

- Monte Carlo는 모나코에 위치한 **‘도박 도시’**
- 스타니스와프 울람(Stanisław Marcin Ulam)이란 미국 수학자가 제안
- **지금 내가 가지고 있는 카드 가지고 카드게임 에서 이길 수 있을 확률은 어느 정도인가**



1. Origin of Monte Carlo Method

- **‘카드게임 에서 이길 수 있을 확률은 어느 정도인가’**

  를 알 수 있는 가장 좋은 방법

1) Deterministic
   - ✓ **모든 경우에 대하여** 각각의 확률을 전부 계산
   - ✓ 확률을 100% 정확하게 계산 가능
   - ✓ 단, 엄청난 수의 조합 때문에 계산 복잡도가 매우 크다.

2) Probabilistic
   - ✓ **무작위의 상황을 만들고 여러 번 simulation**을 거쳐서 통계를 낸다.
   - ✓ 확률은 100% 정확하지 않을 수 있다.
   - ✓ 하지만 근사하게, 비교적 빠르고 쉽게 구할 수는 있다.

1. Origin of Monte Carlo Method

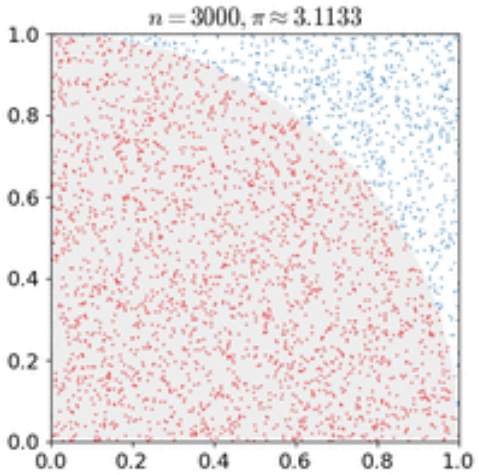
- 즉, Monte Carlo Method란
  - → **'random variable'을 이용하여 어떤 함수의 값을 '확률적'으로 구하는 것을 뜻한다.**
  - → **Error가 발생할 확률이 어느정도 존재하나, 알고리즘은 빠르게 수행 가능**

- **기본 알고리즘**
  1) **무작위로 random variable 생성**
  2) **이를 이용해 구하고자 하는 정보의 확률 계산**
  3) **Variable의 수가 충분히 많아지면 확률 계산 시 실제 확률값으로 근사하게 된다.**

1. Origin of Monte Carlo Method

ex) 원주율(π) 구하기

1) 한 변의 길이가 '$2r$'인 정사각형 내부에 지름이 '$2r$'인 원을 정사각형에 내접하도록 그린다.
2) 이 때, 원과 정사각형의 넓이 비율은 '$\pi r^2 : 4r^2 = \pi : 4$'이다.
3) 정사각형 내부에 **random으로 충분한 수**의 점을 찍고, '**원 내부의 점 : 전체 점**'의 개수 비율을 기록한다.
4) 점의 개수 ≃ 넓이라고 보면 비례식 '$\pi : 4$ = **원 내부의 점 : 전체 점**'을 통해 π 계산.



2. Sampling

- **Sample : 모집단의 '부분집합'**
- **Sampling된 데이터에서 모집단 분포를 추측?**
  - 우선 모든 sampling된 데이터는 **제대로 모집단에서** 추출됨을 가정
  - Random variable: **x** | 확률밀도함수: p
  - Sample 모음: $\boldsymbol{x}$ (p에서 추출) | sample: $\boldsymbol{x}^{(i)}$ | 출력: $f(\boldsymbol{x}^{(i)})$ | 출력평균: $\mathrm{E}[f(\mathbf{x})] = \boldsymbol{s}$

1) n개 sample들을 모아서 출력평균을 구하면 →. $\hat{s}_n = \frac{1}{n}\sum_{i=1}^{n} f(\boldsymbol{x}^{(i)}).$

2) n이 무한이면 sample들의 평균은 모집단의 → 평균으로 수렴한다.

$$\mathbb{E}[\hat{s}_n] = \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}[f(\boldsymbol{x}^{(i)})] = \frac{1}{n}\sum_{i=1}^{n} s = s.$$

$$\lim_{n\to\infty} \hat{s}_n = s,$$

2. Sampling

3) 같은 원리로 $f(x)$의 분산 : $\mathrm{Var}[f(x)]$ 일 때 → 분산은 **모집단 분산/n** 으로 수렴한다.

$$\mathrm{Var}[\hat{s}_n] = \frac{1}{n^2}\sum_{i=1}^{n}\mathrm{Var}[f(\mathbf{x})] = \frac{\mathrm{Var}[f(\mathbf{x})]}{n}.$$

- 즉, **Sampling을 충분히 하면** 모집단의 평균, 분산도 구할 수 있다.
- 꼭 p말고도 sampling에 더 적합한 분포가 있다면 바꿀 수 도 있다.

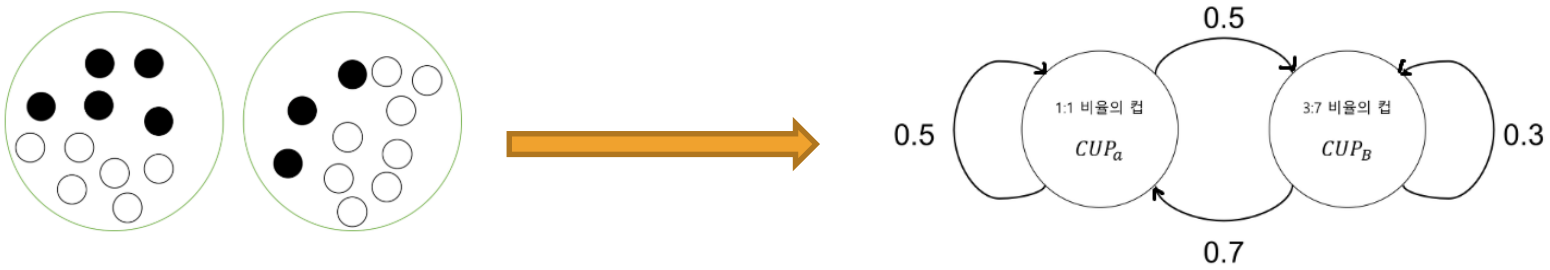
(importance sampling)

$$p(\boldsymbol{x})f(\boldsymbol{x}) = q(\boldsymbol{x})\frac{p(\boldsymbol{x})f(\boldsymbol{x})}{q(\boldsymbol{x})}$$

3. Markov Chain Monte Carlo Method

- **Markov chain이란 random variable이 어떤 state에 도달할 확률은 이전 시점의 state의 영향을 받는 경우를 뜻한다**

- **콩 돌리기 (state 변화 X)**



왼쪽 컵에서 **흰콩**이 나오면 다시 콩을 넣고 왼쪽 컵에서 뽑는다.
왼쪽 컵에서 **검은콩**이 나오면 다시 콩을 넣고 오른쪽컵에서 뽑는다.
오른쪽컵에서 **검은콩**이 나오면 다시 콩을 넣고 오른쪽 컵에서 뽑는다.
오른쪽컵에서 **흰콩**이 나오면 다시 콩을 넣고 왼쪽 컵에서 뽑는다.

3. Markov Chain Monte Carlo Method

- **만일, 시행에 따라 state가 계속 변하게 되는 경우**

  ex) 날씨 예측 (‘X_k = 맑음 or 비’)

  ① state - P(맑음|맑음) = 0.9

  - P(맑음|비) = 0.1

  - P(비|맑음) = 0.7

  - P(비|비) = 0.3

  ① State를 통해 다음 state를 예측하고, 예측한 state로 그 다음 state를 예측 반복

  ② 이 과정을 반복하면 나중에 **현재 날씨확률은 전날 날씨 확률과 같아진다.**

  ③ 즉, sampling 관점으로 보면, 현재 sample이 이전 sample의 영향을 받는다.

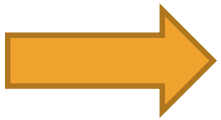
3. Markov Chain Monte Carlo Method

- State의 변화를 행렬로 만들면 아래와 같이 되며 이를 transition probability 라고 한다.

$$\mathbf{A} = \begin{bmatrix} .9 & .1 \\ .3 & .7 \end{bmatrix}$$

$$X_0 = [.1, \ .0]$$

(오늘은 맑음)



$$X_1 = X_0 * \mathbf{A} = [.1 \ .0] \begin{bmatrix} .9 & .1 \\ .3 & .7 \end{bmatrix} = [\mathbf{.9} \ \mathbf{.1}]$$

$$X_2 = X_1 * \mathbf{A} = [.9 \ .1] \begin{bmatrix} .9 & .1 \\ .3 & .7 \end{bmatrix} = [\mathbf{.84} \ \mathbf{.16}]$$

$$X_3 = X_2 * \mathbf{A} = [.84 \ .16] \begin{bmatrix} .9 & .1 \\ .3 & .7 \end{bmatrix} = [\mathbf{.804}, \ \mathbf{.196}]$$

.

.

$$X_k = X_0 * \mathbf{A}^k$$

$$= X_{k-1} * \mathbf{A}$$

$$= X_{k-1}(\text{수렴})$$

- 이 과정을 거치면 최후에는 $X_k = X_{k-1}$, 즉 stationary distribution으로 수렴하게 된다 (단, 항상 수렴하지는 않는다).

3. Markov Chain Monte Carlo Method

◆ 단점도 존재

- **현재 sample과 decorrelated된 sample 뽑으려면 일정 시간 이상 기다려야 한다.**

  → 즉, sample당 시간 = 수렴 + decorrelation.

  → 그래서 보통 다수의 Markov chain을 돌려서 사용한다.

- **또한 모집단 형태로의 실제 수렴(mixing)까지 얼마나 걸리는 지도, 도달 했는지 아닌지 판단기도 힘들다.**

- **실제 상황에서는 state자체도 매우 많아서 입력 및 A 설정도 하기도 힘들다.**

4. Gibbs Sampling

- **n개의 random variable로 이루어진 sample을**
- **Markov chain을 유지한 채**
- **k개 추출하려면?**

n개의 확률변수 $(X_1, \cdots, X_n)$의 결합 확률 분포 $p(x_1, \cdots, x_n)$로부터 k개의 표본 X를 얻으려고 할 때, 기브스 표집은 다음과 같이 동작한다.

1. 임의의 $X^{(0)} = (x_1^{(0)}, \cdots, x_n^{(0)})$을 선택한다.
2. 각 변수 $x_1, \cdots, x_n$에 대하여, 현재의 값을 기반으로 한 새로운 값을 조건부 확률분포 $p(x_i^{(t)} | x_1^{(t)}, \cdots, x_{i-1}^{(t)}, x_{i+1}^{(t-1)}, \cdots, x_n^{(t-1)})$에서 표집한다.
3. $X^{(t)} = (x_1^{(t)}, \cdots, x_n^{(t)})$를 X에 추가한다.

현재 t번째 Sample 구성요소 중 i번째 변수

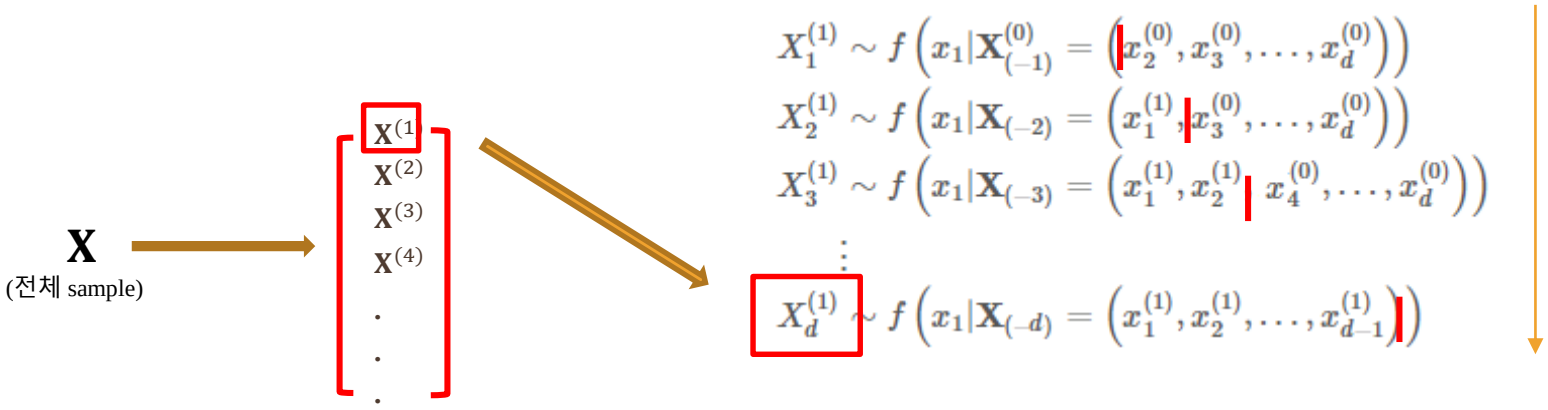
현재 Sample 구성요소로 먼저 뽑힌 변수들

아직 현재 Sample로 뽑히지 않은 이전 Sample의 변수들

i 빼고 조건에 포함 $\mathbf{X}_{(-i)}$

4. Gibbs Sampling

- t번째 sample $\mathbf{X}^{(t)}$는 보다시피 $X_1^{(t)}, X_2^{(t)}, X_3^{(t)}, X_4^{(t)}$ … 이런 식으로 update 되면서 Markov chain을 이루게 된다. Chain이 수렴되면 다음 sample $\mathbf{X}^{(t+1)}$의 Markov chain을 시작.



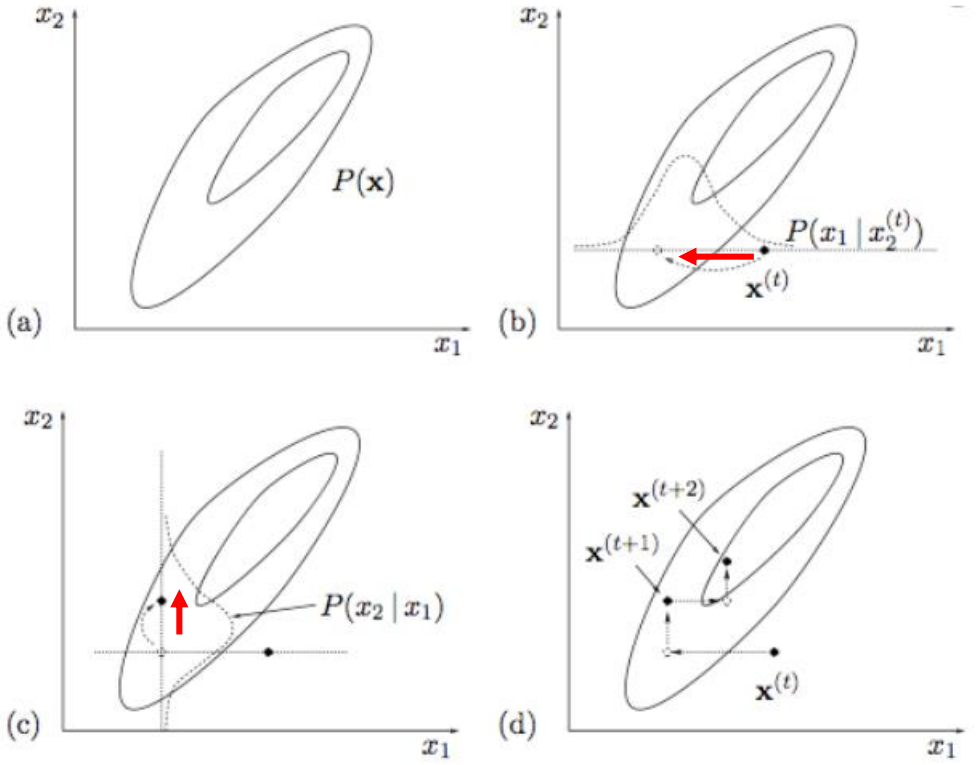
- $\mathbf{X}^{(0)}$ (초기 sample)은 임의의 값으로 설정해도 chain이 수렴하면 초기값에 관계없이 p에 기반한 sample을 얻을 수 있다. 그래서 $\mathbf{X}^{(0)}$는 sample로 쓰이지 않고 버려진다.

4. Gibbs Sampling

EX) $X^{(t)} = P(x_1, x_2)$, 즉 X가 2차원으로 이루어져 있을 때,

1) t+1번째 sample의 x_1 → 이전 sample x_2를 조건으로 하여 update → $P(x_1 \mid x_2^{(t)})$

2) t+1번째 sample의 x_2 → x_1을 조건으로 하여 update → $P(x_2 \mid x_1)$



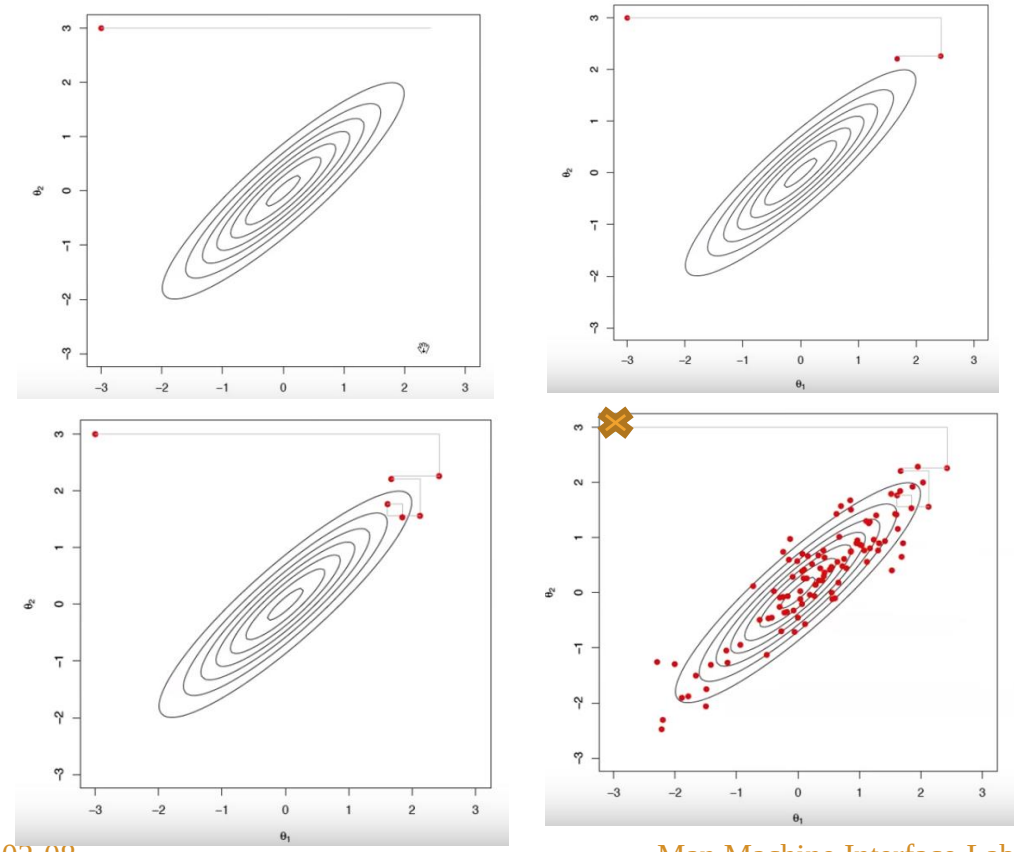
↔ : sample의 성분 중 $\boldsymbol{x_1}$ update

↕ : sample의 성분 중 $\boldsymbol{x_2}$ update

4. Gibbs Sampling

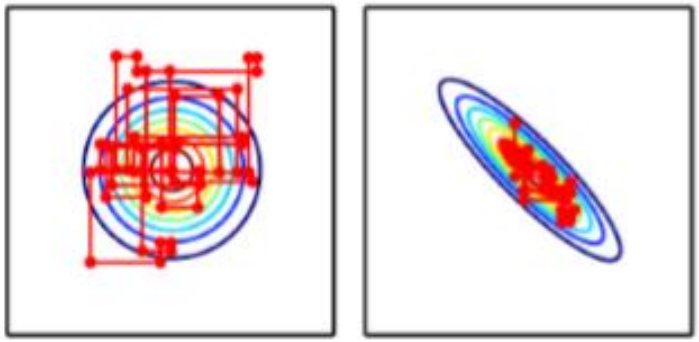
EX) $X^{(t)} = P(x_1, x_2)$, 즉 X가 2차원으로 이루어져 있을 때,

1) x_1은 이전 sample x_2를 조건으로 하여 update
2) x_2는 x_1을 조건으로 하여 update



↔ : sample의 성분 중 x_1 update
↕ : sample의 성분 중 x_2 update

4. Gibbs Sampling

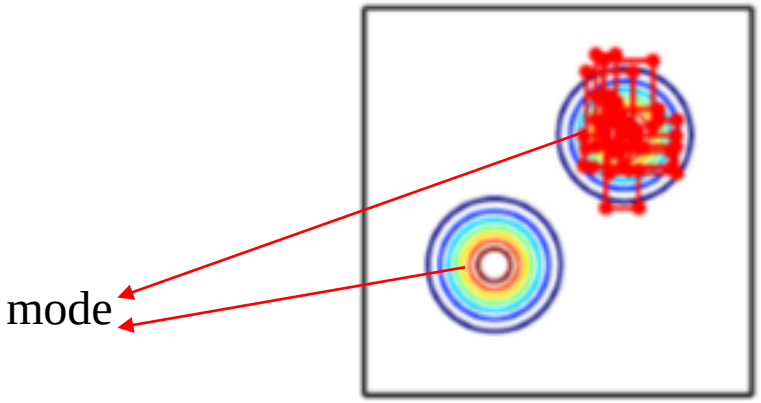


- **왼쪽은 2개의 variable이 서로 independent하여 Gibbs sampling시 mixing이 잘된다.**
  - → **즉, sample들이 바깥에서 차츰 안쪽으로 뽑히고 그 sample들은 왼쪽과 같은 분포를 가진다.**

- **오른쪽 2개의 variable이 서로 highly correlated 되어 있어 mixing이 느리다(심지어 안되기도).**
  - → Sample의 분포가 distribution의 일부에만 몰려 있다.
  - → Correlation이 클수록 step size가 작아져서 작은 범위 내에서만 맴돌게 되어 전체 분포를 나타내지를 못한다.

5. Mixing between separated modes

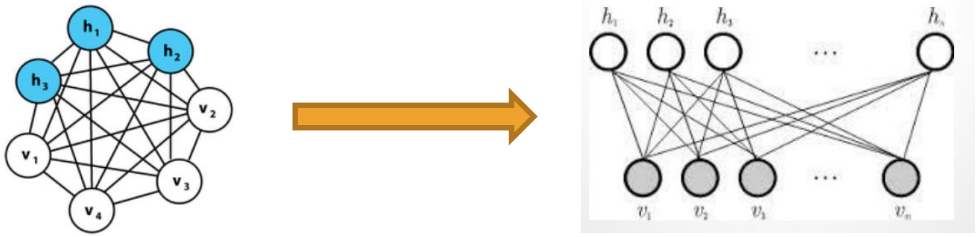
- **Mixture of Gaussian같이 mode가 2개 있는 경우도 있다.**
- **전체 sample 집합내에는 각 mode에서 가져온 sample들이 모집단과 비슷한 비율로 존재해야 한다.**



- **이를 manifold learning에 적용 시키면? → $p(h, x)$**
  - → Generative model에서 $p(h \mid x)$ → **encoding,** $p(x \mid h)$ → **decoding**
  - → Classification에서 class별로 존재하는 manifold가 위처럼 분리되어 있으면, **한쪽 mode의 manifold에서만 sampling 될 수도 있고, 양쪽 mode에서 sampling 되더라도 mixing이 느릴 수 밖에 없다.**

6. Tempering to Mix between modes

※ RBM : 제한된 볼츠만 머신. Visible layer와 hidden layer 2개로 이루어져 있다.
기존 볼츠만 머신에서 같은 layer내 unit끼리 학습에 영향 주는 것을 제한



※ 에너지 : 기본적으로 단층이기 때문에 error 만큼의 backpropagation이 존재하지 않는다. 그래서 대신 도입한 개념.
입력과 결과가 같을 수록 값이 작아진다.

$$Energy(\mathbf{x}, \mathbf{h}) = -\mathbf{b}'\mathbf{x} - \mathbf{c}'\mathbf{h} - \mathbf{h}'\mathbf{W}\mathbf{x}$$

※ 볼츠만 머신은 이처럼 에너지 기반 모델(Energy based model) 중 하나로, 에너지 기반 모델에서는 에너지 함수를 통해 확률 분포를 다음과 같이 정의한다.

$$P(x) = \sum_{\mathbf{h}} P(x, h) = \sum_{\mathbf{h}} \frac{e^{-Energy(x,h)}}{Z}$$

6. Tempering to Mix between modes

- **해결 방법은?**
  - → Mixing이 잘 되기 위해서는 **distribution의 peak가 작아야 한다 (in 제한된 볼츠만 머신).**
  - → **Generative model에서는 h와 x가 서로 highly mutual해야 한다. Distribution으로 따지면 겹치는 영역이 크다는 뜻. 그런데 한쪽의 peak가 너무 커버리면 mutuality가 감소.**
  - → **볼츠만 머신에서는 '에너지'에 따라 peak가 결정되므로 에너지에 extra parameter를 추가.**

$$p(\boldsymbol{x}) \propto \exp\left(-E(\boldsymbol{x})\right)$$

$$p_\beta(\boldsymbol{x}) \propto \exp\left(-\beta E(\boldsymbol{x})\right)$$

  - → **Tempering이란 바로 $\beta < 1$로 설정하여 sampling하면서 mode를 mixing하는 것을 말한다.**

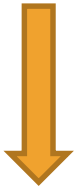
6. Tempering to Mix between modes

- 왼쪽 : Gibbs sampling을 이용하여 뽑은 successive sample
  - → Mixing이 제대로 되지 않아 sample이 제대로 생성되지 않은 모습이다.
  - → sample간 의존도도 높아 비슷한 sample만 생성.

- 오른쪽 : ancestral sampling을 이용하여 뽑은 successive sample
  - → sample간에는 독립적이라 mixing 관련 문제가 존재하지 않는다..

7. Depth may help mixing

- $p(h, x)$에서 $p(h \mid x)$가 x를 너무 잘 encode하면 $p(x \mid h)$가 decoding시 ***x*에서 크게 벗어나지를 못한다.**
- **이는 autoencoder, 또는 RBM등을 deep하게 쌓으면 해결이 된다.**



- **Reconstruction error는 감소.**
- **Representation 학습하는 h-space 증가로 인해 h의 분포가 *x*보다 더 넓어지게 하는 효과.**
- **Mode간의 gap 감소**