Dicing

Chen Yunchuan
2015.4.29

Outline

- Sampling
    - Random Variable Generation
    - Rejection Sampling
        - Importance Sampling
        - Markov Chain Monto Carlo (MCMC)
- Restricted Boltzmann Machine Revisit
    - Learning algorithms
- Imagination: Represent Word with Random Vectors

Example of Monte Carlo: Estimate π

procedure `direct-pi`
$N_{\text{hits}} \leftarrow 0$ (initialize)
for $i = 1, \ldots, N$ **do**

$$\begin{cases} x \leftarrow \texttt{ran}\,(-1, 1) \\ y \leftarrow \texttt{ran}\,(-1, 1) \\ \textbf{if } (x^2 + y^2 < 1)\ N_{\text{hits}} \leftarrow N_{\text{hits}} + 1 \end{cases}$$

output N_{hits}

- Estimate π

$$\pi \approx \frac{4N_{\text{hits}}}{N}$$

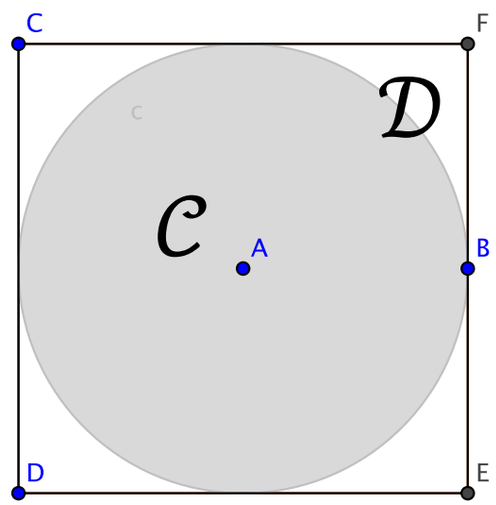
Run	N_{hits}	Estimate of π
1	3156	3.156
2	3150	3.150
3	3127	3.127
4	3171	3.171
5	3148	3.148

Estimating π

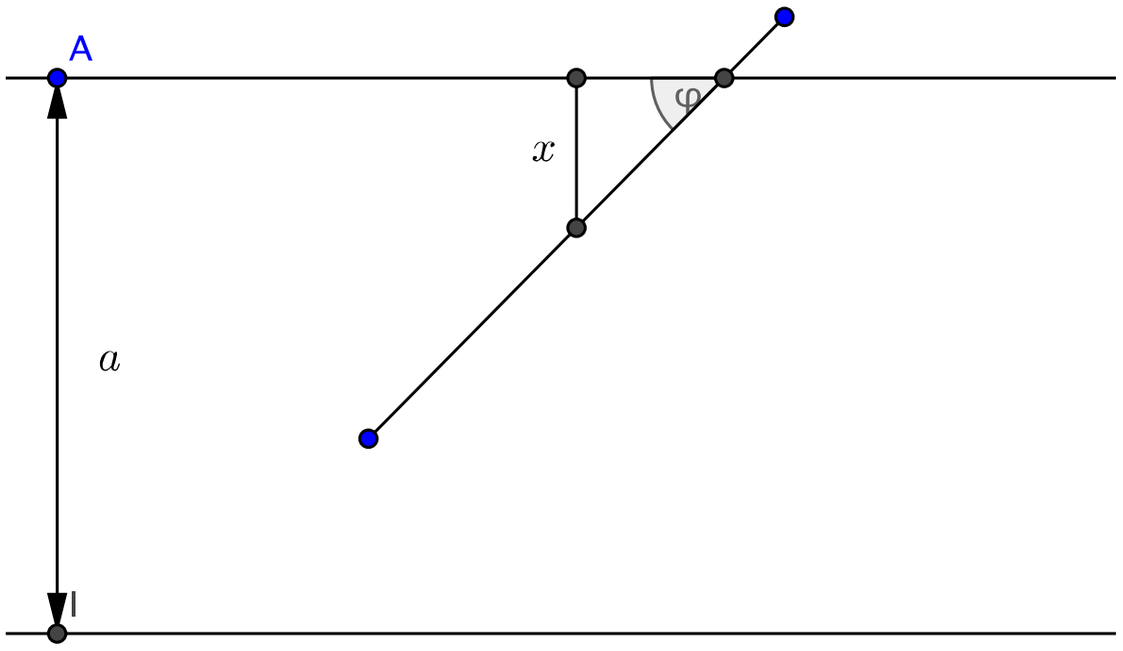
Draw i.i.d. set of samples $\{\boldsymbol{x}^{(i)}\}_{i=1}^{N}$
from a target density $p(\boldsymbol{x}|\cdot)$

$$\int_{\mathcal{X}} f(\boldsymbol{x})p(\boldsymbol{x}|\cdot)\mathrm{d}\boldsymbol{x} \approx \frac{1}{N}\sum_{i=1}^{N} f(\boldsymbol{x}^{(i)})$$

$$\begin{aligned}
\frac{\pi}{4} &= \frac{S_{\mathcal{C}}}{S_{\mathcal{D}}} = \frac{\iint_{\mathcal{C}} \mathrm{d}x\mathrm{d}y}{\iint_{\mathcal{D}} \mathrm{d}x\mathrm{d}y} = \frac{\iint_{\mathcal{C}} p(x,y)\mathrm{d}x\mathrm{d}y}{\iint_{\mathcal{D}} p(x,y)\mathrm{d}x\mathrm{d}y} \\
&= \iint_{\mathcal{D}} \mathbb{1}\{(x,y)\in\mathcal{C}\}p(x,y)\mathrm{d}x\mathrm{d}y \\
&\approx \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}\{(x^{(i)},y^{(i)})\in\mathcal{C}\} = \frac{N_{\text{hit}}}{N}
\end{aligned}$$

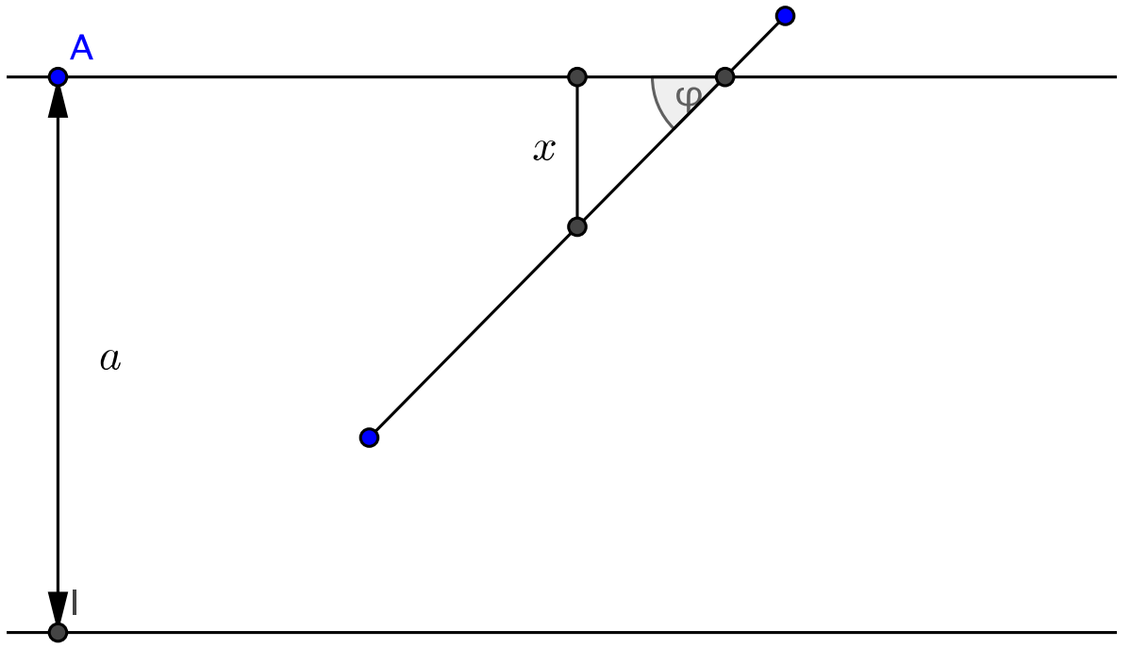


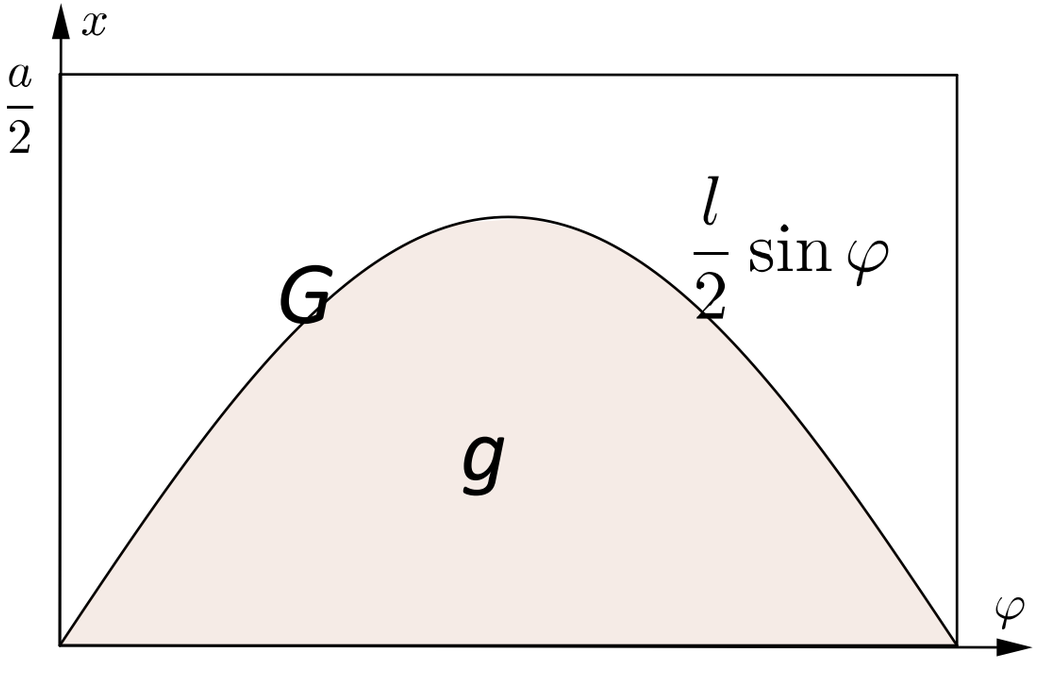
Another Way to Estimate π



$$\pi \approx \frac{2lN}{aN_{\text{hits}}}$$

Estimating π





$$p_{\text{hit}} = \frac{S_g}{S_G} = \frac{\int_0^\pi \frac{l}{2}\sin\varphi \, \mathrm{d}\varphi}{\frac{1}{2}a\pi} = \frac{2l}{\pi a}$$

$$p_{\text{hit}} \approx \frac{N_{\text{hits}}}{N} \qquad \pi \approx \frac{2lN}{aN_{\text{hits}}}$$

Estimating π

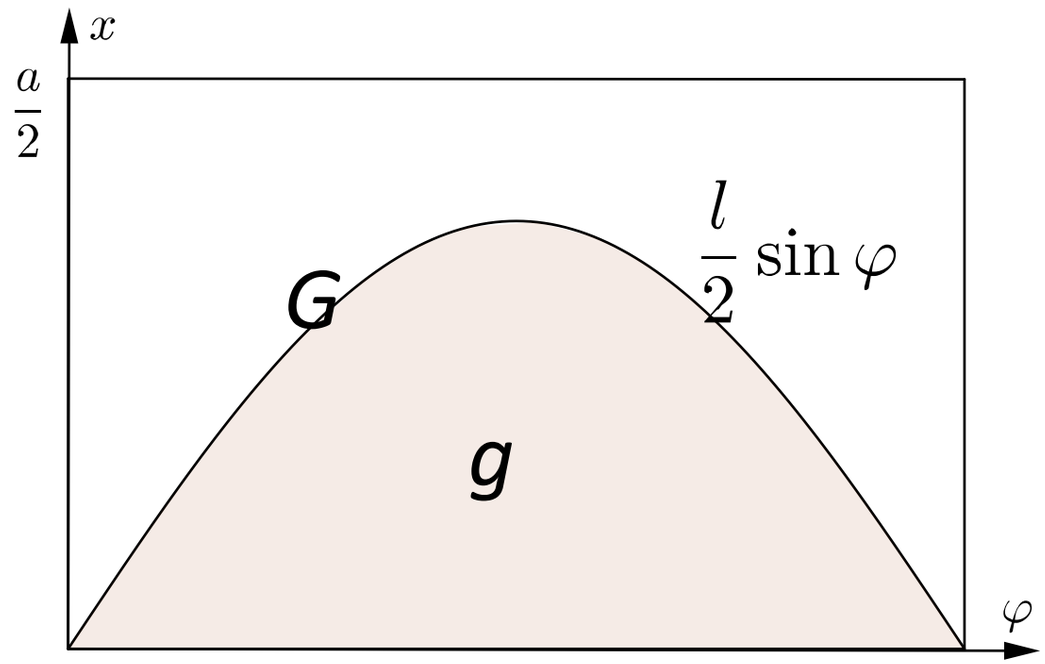
$$\frac{2l}{\pi a} = \frac{S_g}{S_G} = \frac{\iint_g \mathrm{d}x\mathrm{d}y}{\iint_G \mathrm{d}x\mathrm{d}y}$$

$$= \frac{\iint_g p(x,y)\ \mathrm{d}x\mathrm{d}y}{\iint_G p(x,y)\ \mathrm{d}x\mathrm{d}y}$$

$$= \iint_G \mathbb{1}\{(x,y) \in g\} p(x,y)\ \mathrm{d}x\mathrm{d}y$$

$$= \mathbb{E}[\mathbb{1}\{(x,y) \in g\}]$$

$$\approx \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}\{(x,y) \in g\} = \frac{N_{\text{hits}}}{N}$$



Monte Carlo: Motivation

- Estimation
- Optimization
- Model Selection

Random Variable Generation

- if $U \sim \mathcal{U}_{[0,1]}$, then the random variable $F^-(U)$ has the distribution F where $F^-(u) = \inf\{x : F(x) \geq u\}$

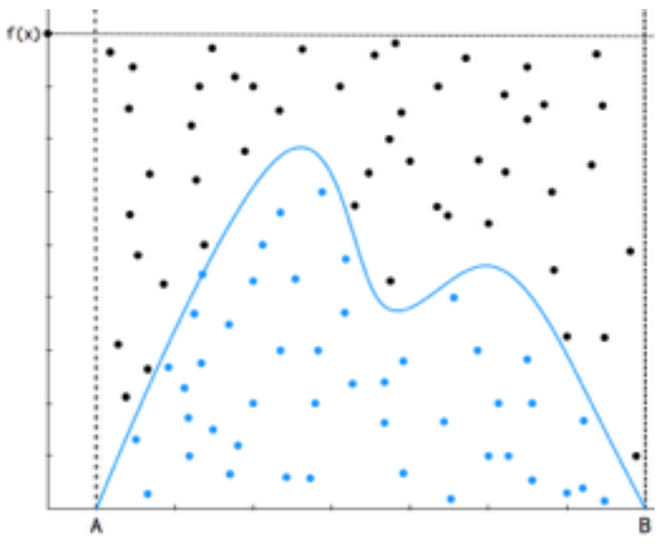
- The theorem above is theoretical beautiful, but unpractical in real application
  - It is hard to compute the generalized inverse function
  - High dimensional cases?

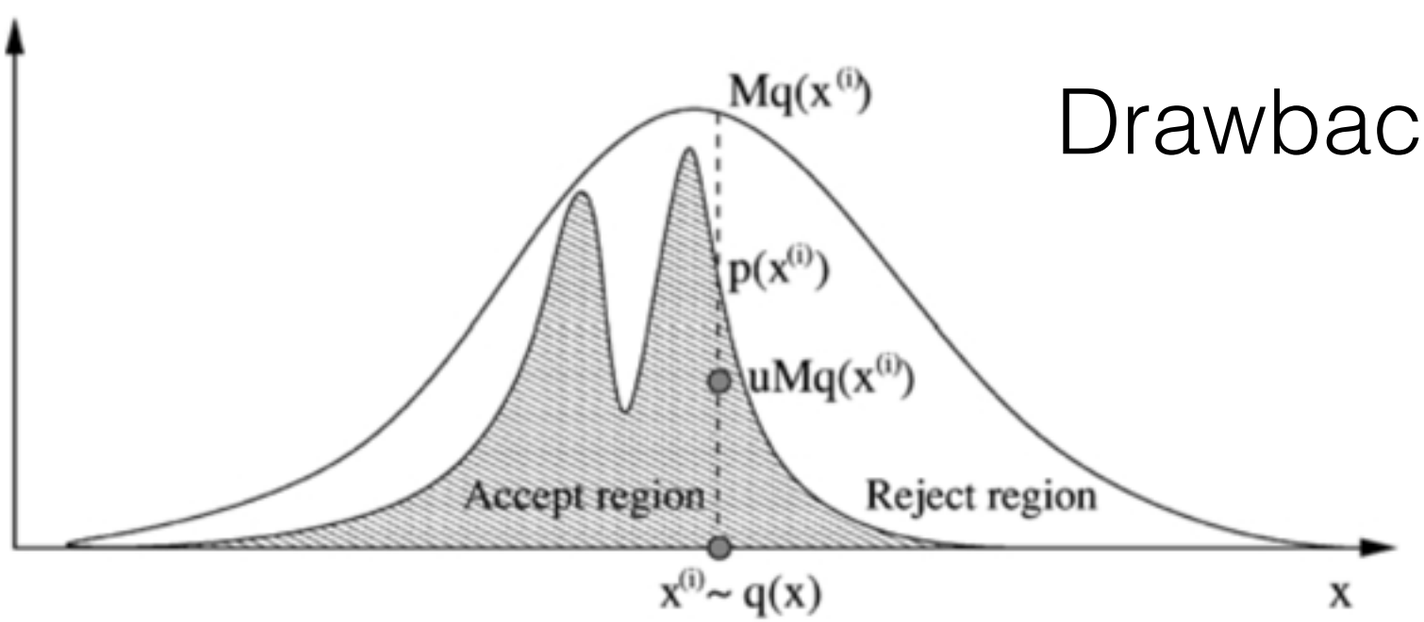
Rejection Sampling

Set $i = 1$

Repeat until $i = N$

1. Sample $x^{(i)} \sim q(x)$ and $u \sim \mathcal{U}_{(0,1)}$.
2. If $u < \frac{p\left(x^{(i)}\right)}{Mq\left(x^{(i)}\right)}$ then accept $x^{(i)}$ and increment the counter i by 1. Otherwise, reject.



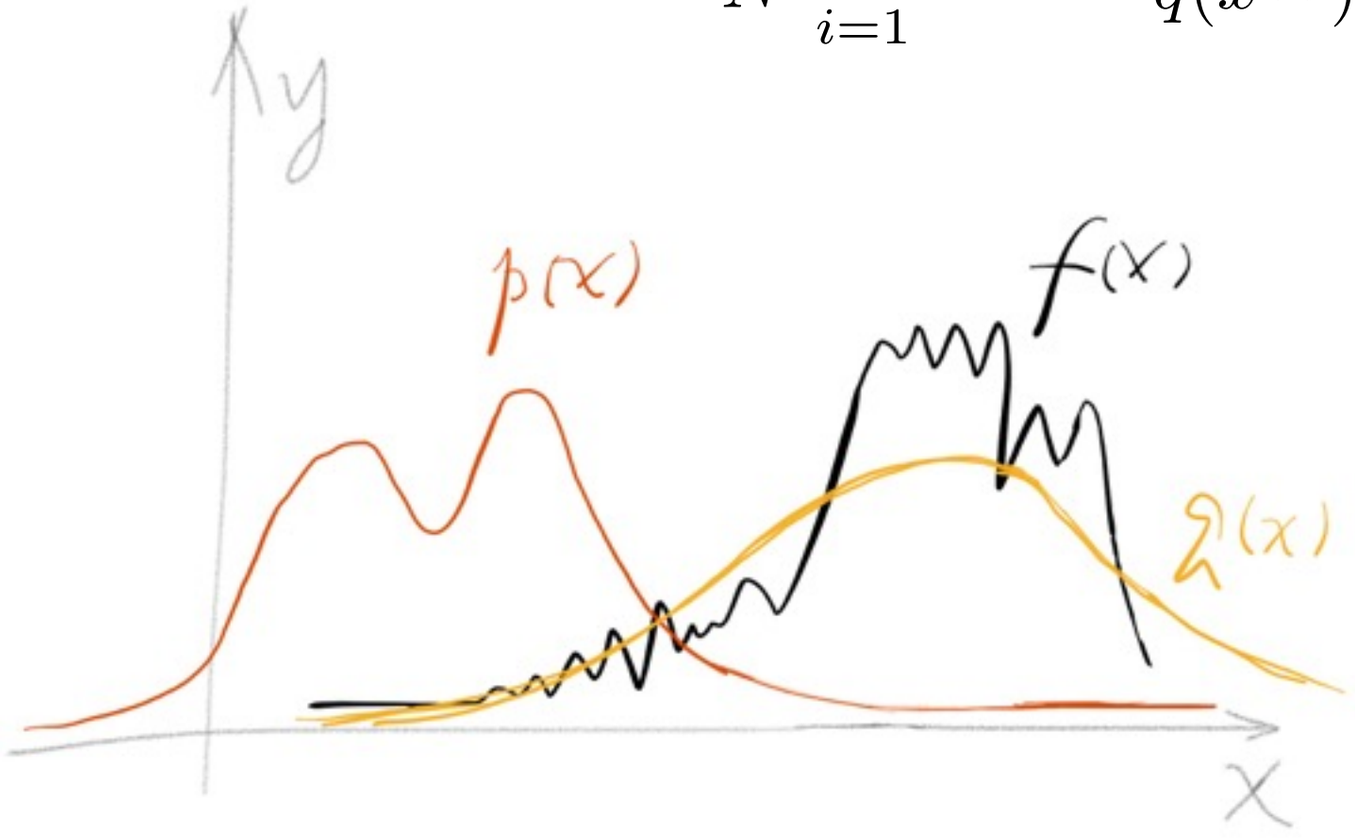


Drawbacks?

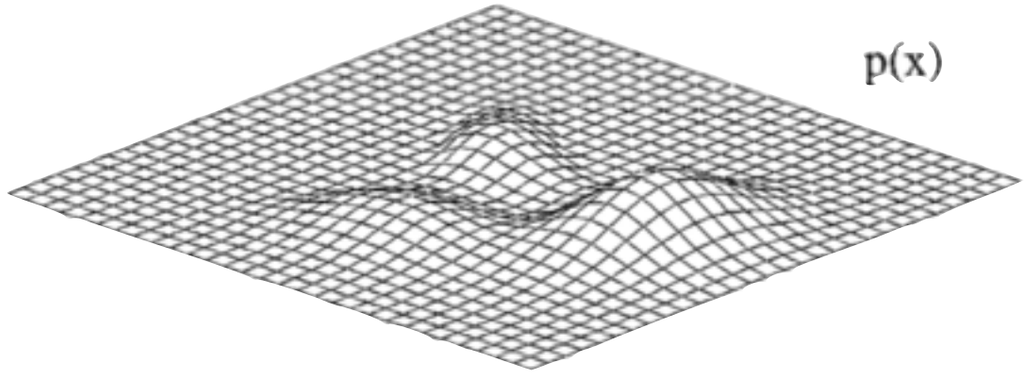
Importance Sampling

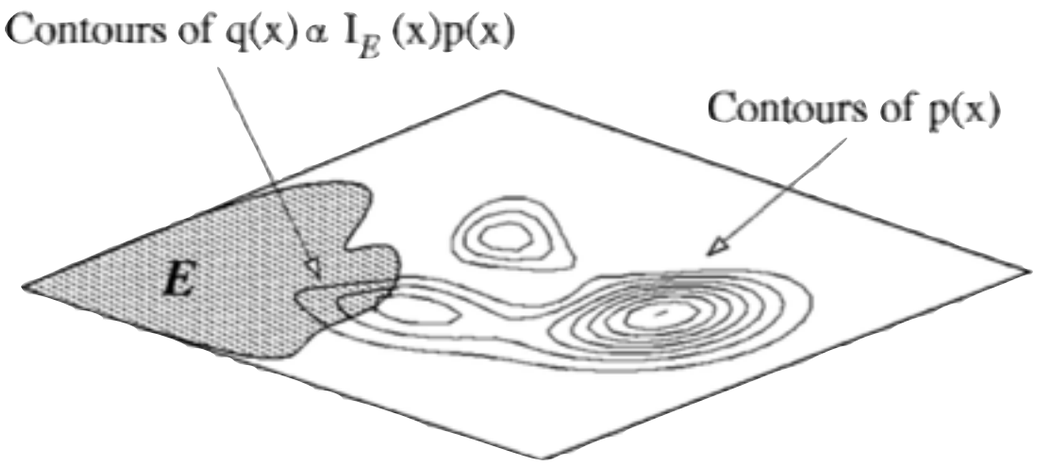
$$\int f(x)p(x)\,\mathrm{d}x = \int f(x)\frac{p(x)}{q(x)}q(x)\,\mathrm{d}x$$

$$\approx \frac{1}{N}\sum_{i=1}^{N} f(x^{(i)})\frac{p(x^{(i)})}{q(x^{(i)})}$$



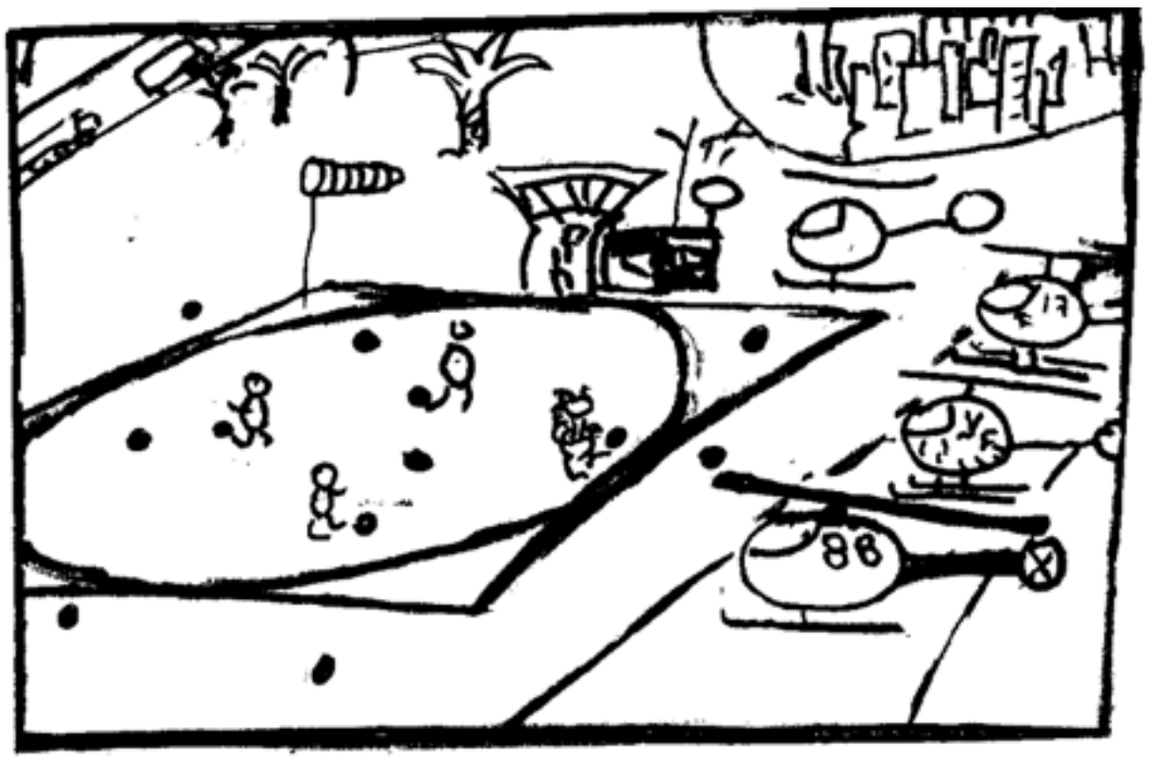
$\mathrm{Var}_p[f(x)] > \mathrm{Var}_q\left[\frac{p(x)}{q(x)}f(x)\right]$

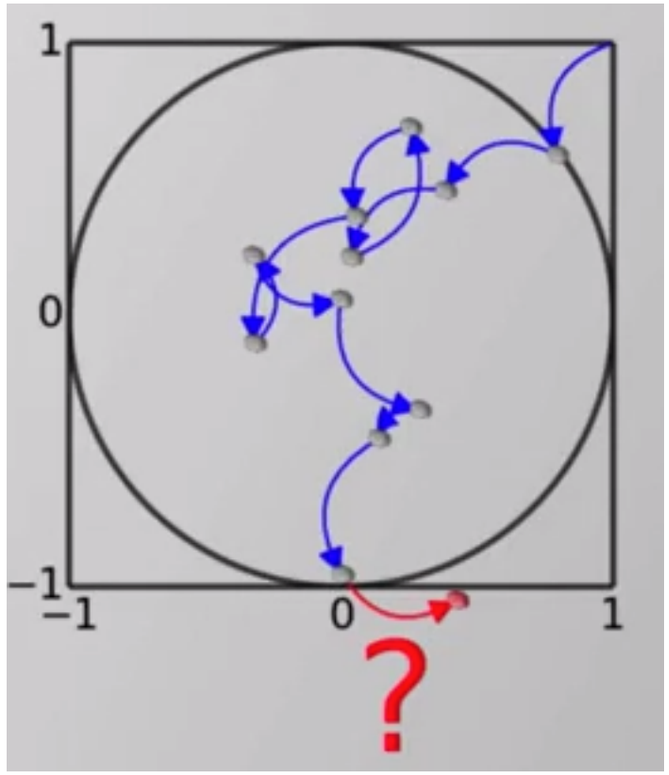




- Does the i.i.d. property necessary for estimating an integral?

Another π Estimation Story

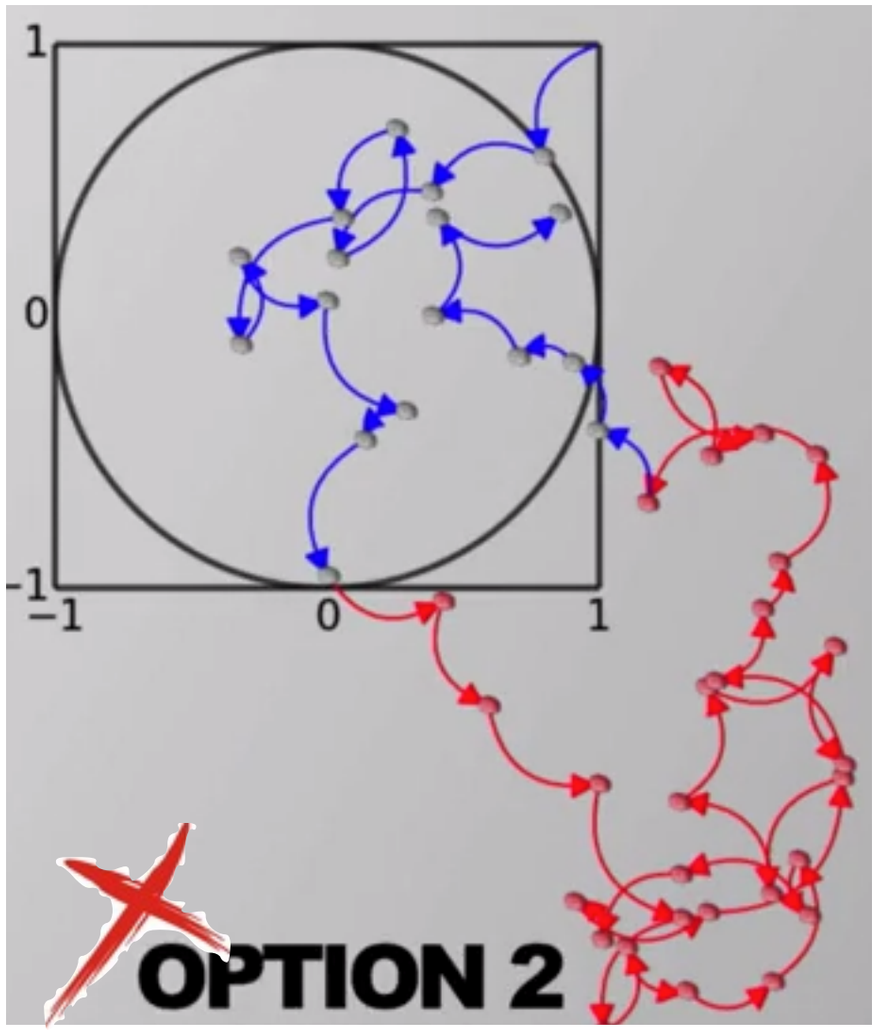


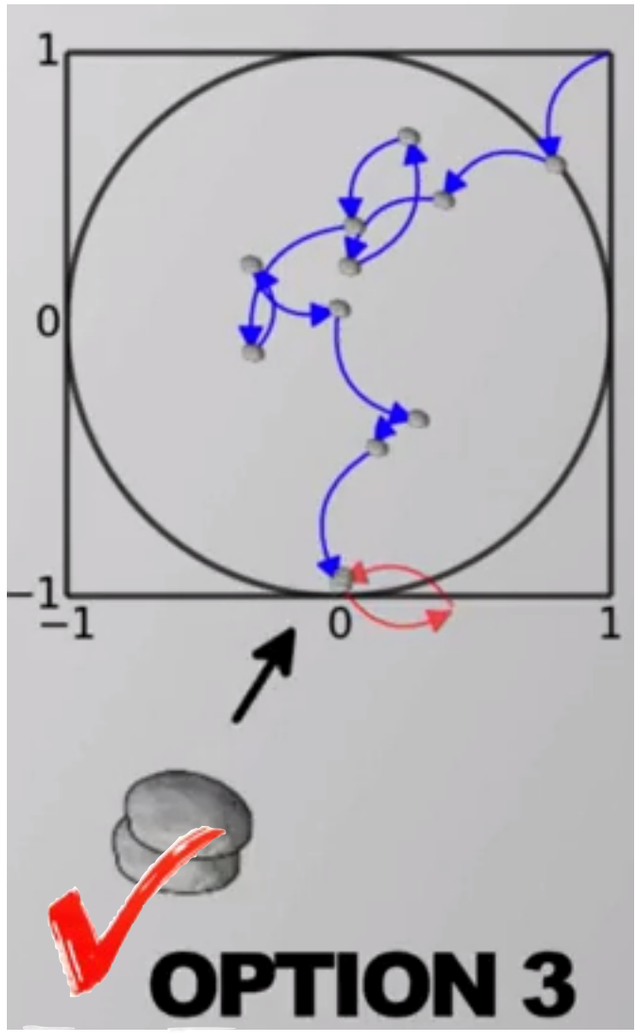


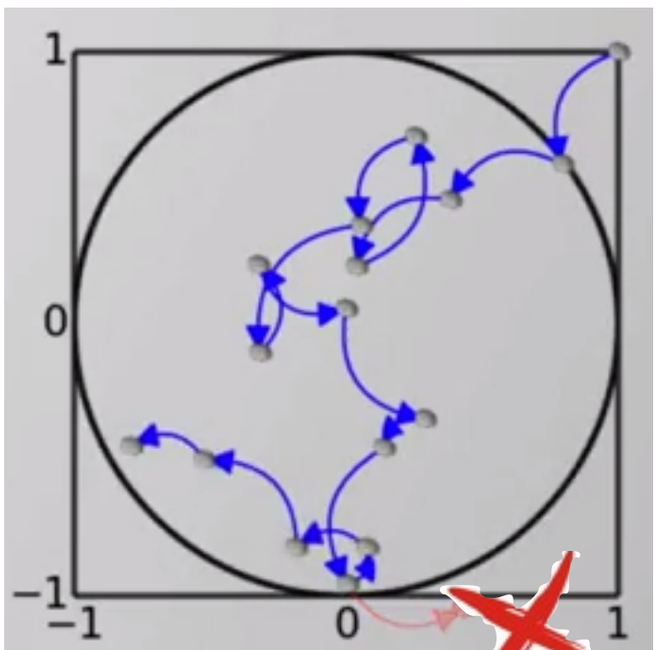
?

?

?



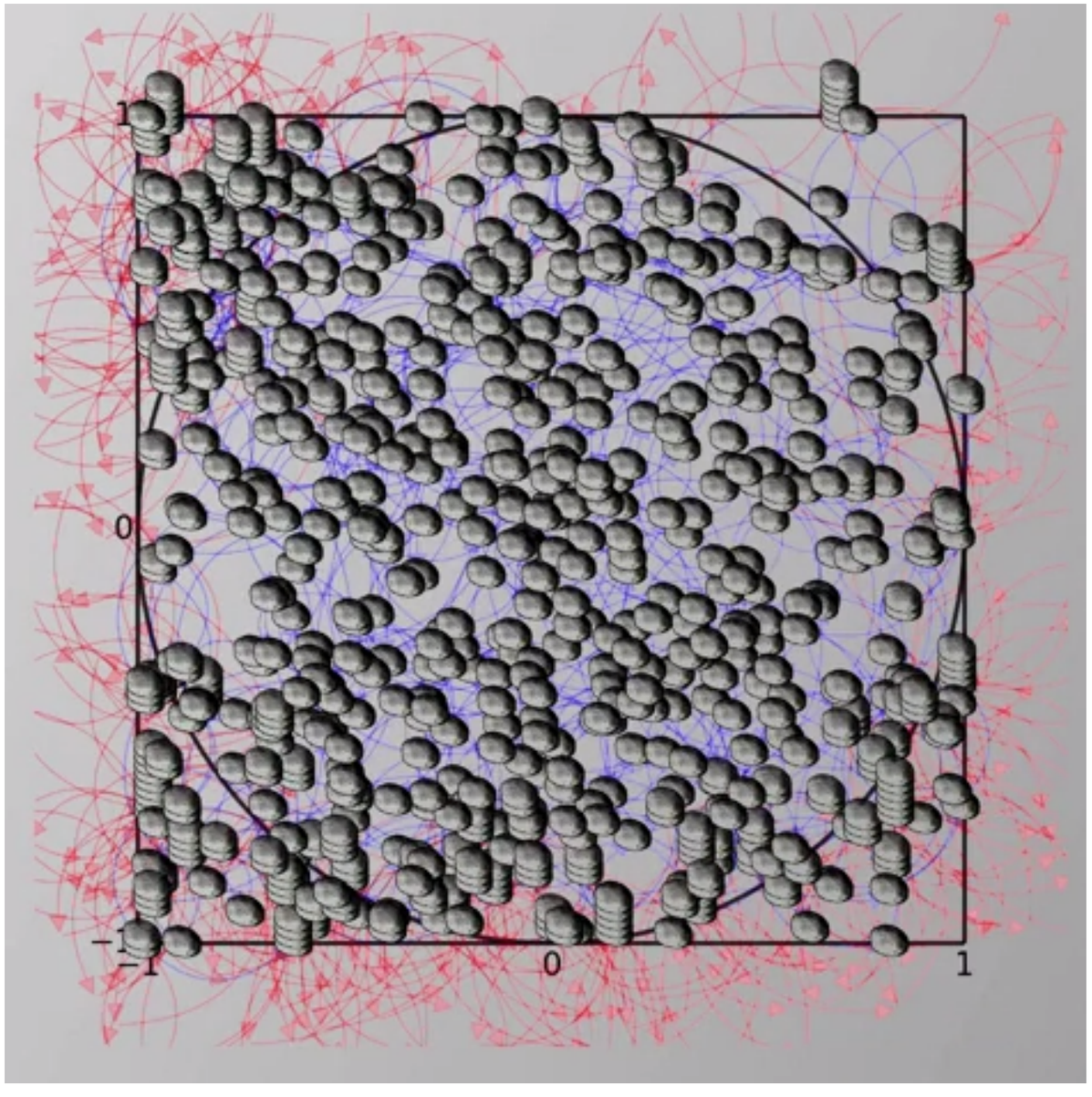


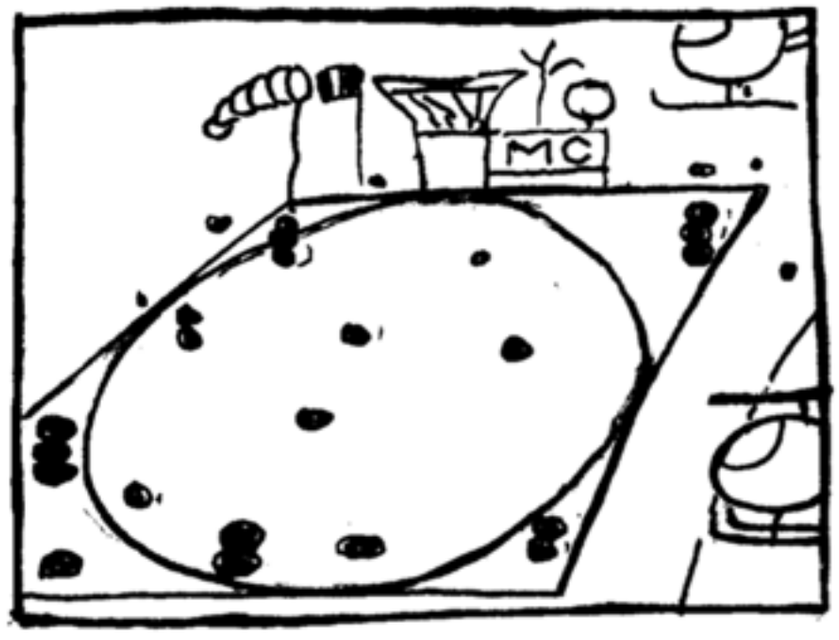


OPTION 1

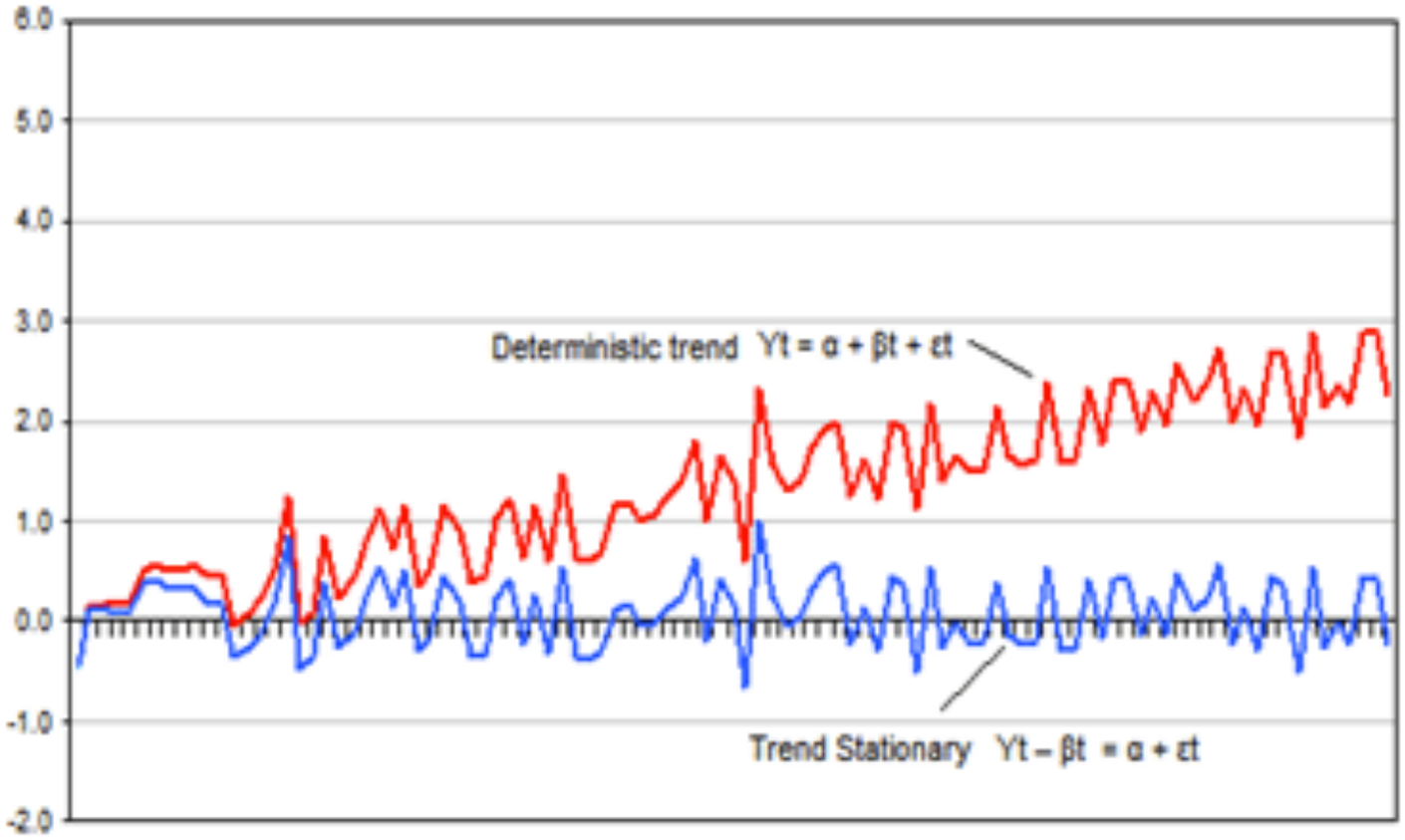
OPTION 2

OPTION 3


0
−1
−1
0
1


MC

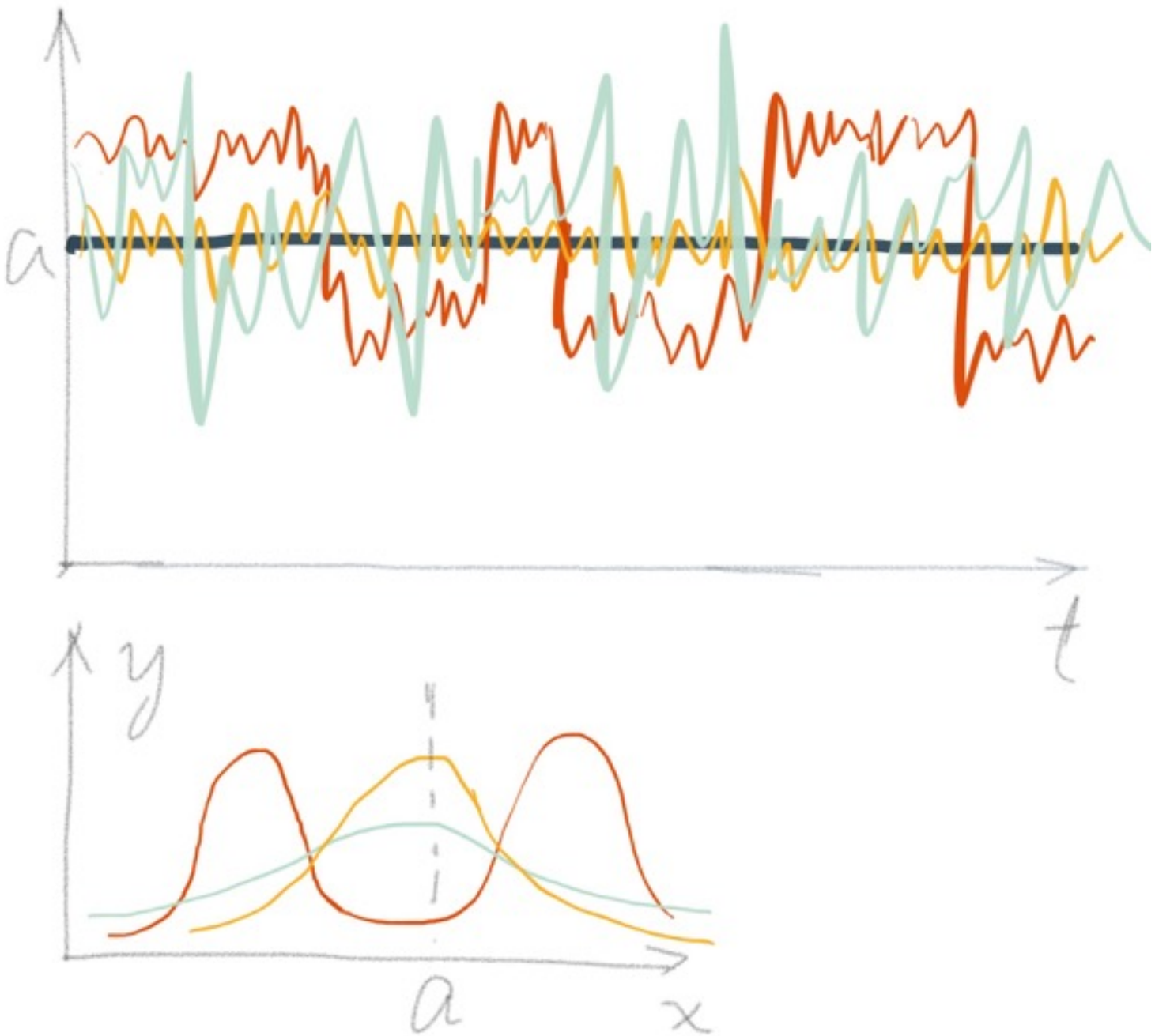
Stationary Stochastic Process



$$\mathbb{E}[X_t] = \text{constant} = \lim_{T \to +\infty} \frac{1}{T} \int_0^T X(t)\, \mathrm{d}t$$

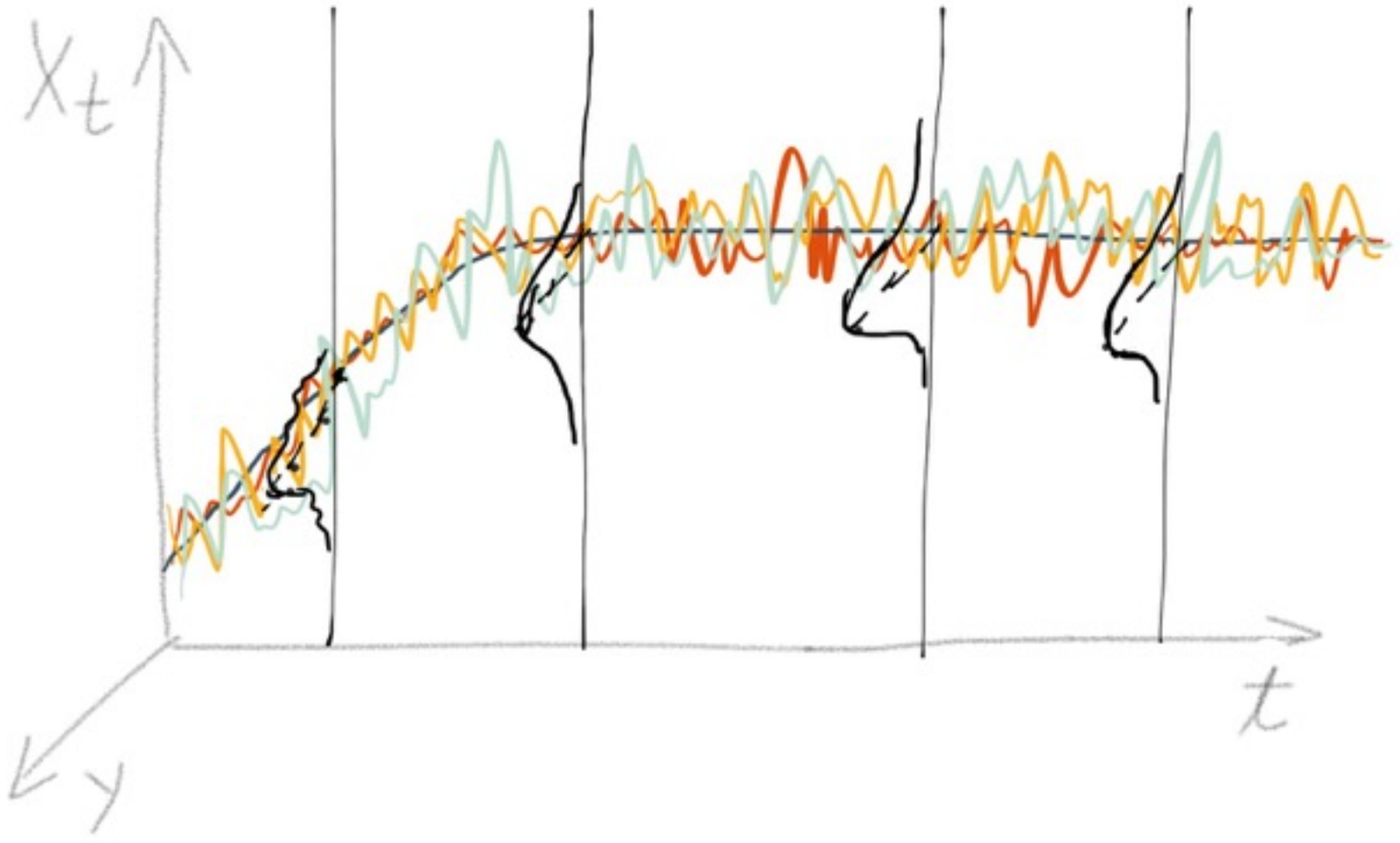
? $$\mathbb{E}[f(X_t)] = \lim_{T \to +\infty} \frac{1}{T} \int_0^T f(X(t))\, \mathrm{d}t$$

Why $\mathbb{E}[f(X_t)] = \lim_{T\to+\infty} \frac{1}{T}\int_0^T f(X(t))\,\mathrm{d}t$?

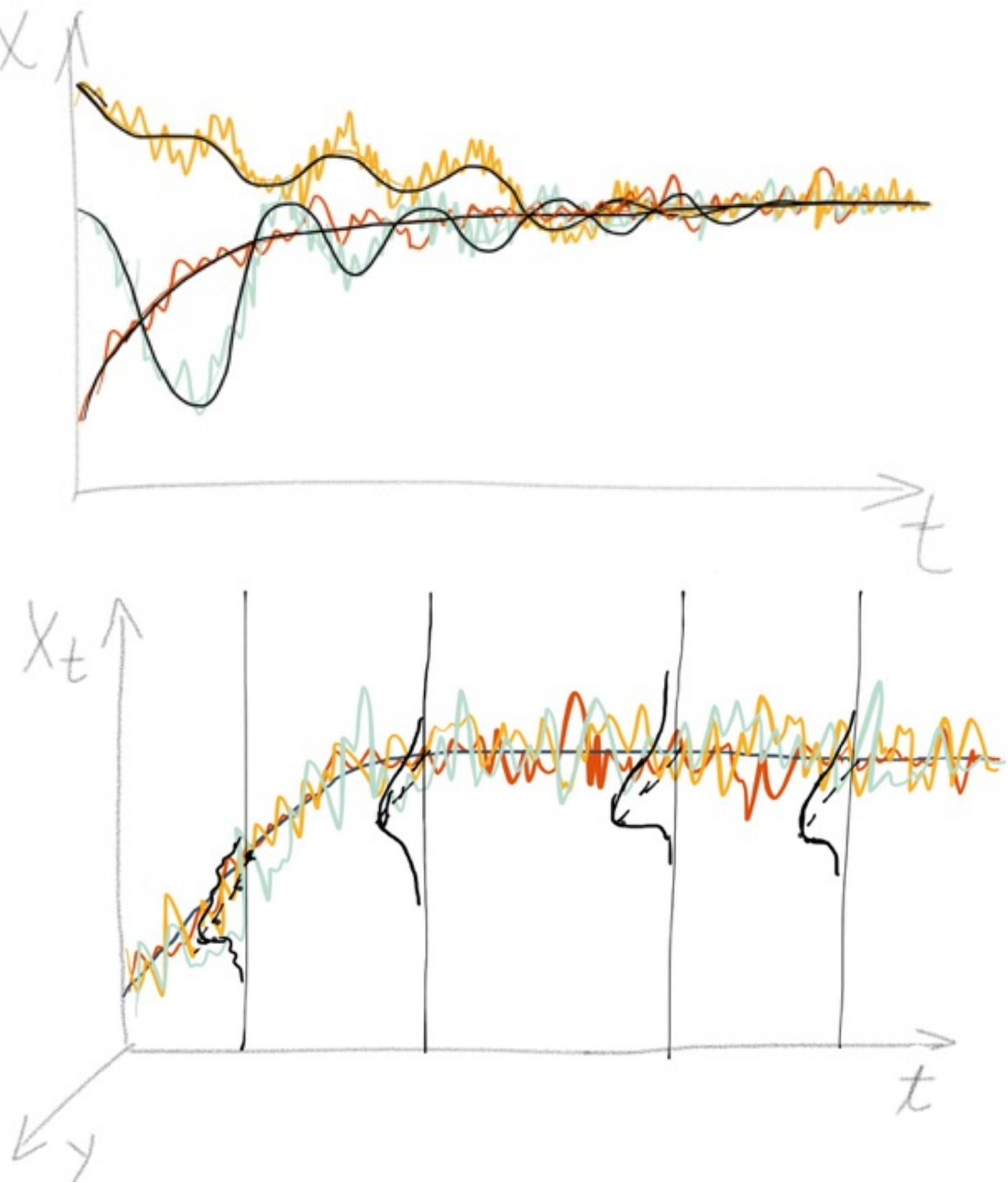


- How to make the equation below hold?

$$\mathbb{E}[f(X_t)] = \lim_{T \to +\infty} \frac{1}{T} \int_0^T f(X(t)) \, \mathrm{d}t$$



Markov Chain Monte Carlo (MCMC): Intuitions

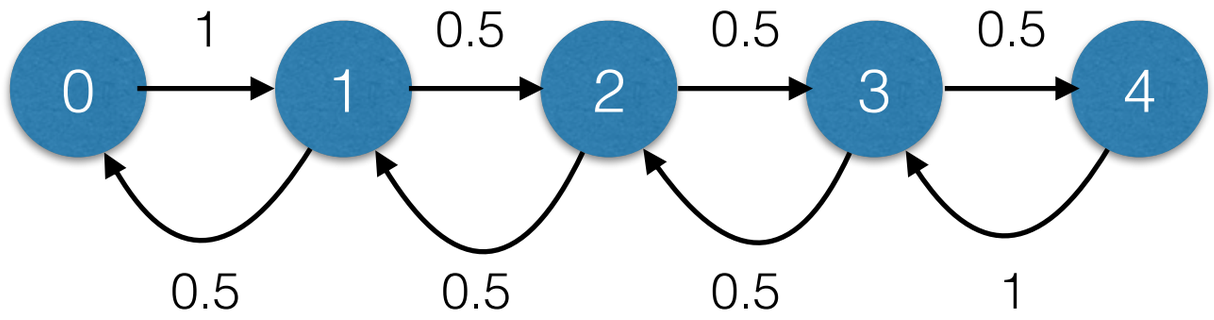


MCMC: Motivation

- Bayesian inference and learning
  - Normalization $$p(x|y) = \frac{p(y|x)p(x)}{\int_{\mathcal{X}} p(y|x')p(x') \, \mathrm{d}x'}$$
  - Marginalization $$p(x|y) = \int_{\mathcal{Z}} p(x, z, |y) \, \mathrm{d}z$$
  - Expectation $$\mathbb{E}_{p(x|y)}[f(x)] = \int_{\mathcal{X}} f(x)p(x|y) \, \mathrm{d}x$$
- Optimization
- Statistical mechanics: Compute the partition function
- Penalized likelihood model selection

MCMC: Important Theoretical Results

- Irreducibility



- Aperiodicity
- Detailed Balance Condition

$$\pi_i P_{ij} = \pi_j P_{ji}$$

MCMC: Applications to Sampling Algorithms: Metropolis-Hastings Alg. (MHA)

- Proposal Distribution

$$q(i,j) = q(x^{(j)}|x^{(i)})$$

- Acceptance Prob.

$$\alpha(i,j) = \mathcal{A}(x^{(i)}, x^{(j)})$$

- Target Distribution

$$\pi(i) = p(x^{(i)})$$

- Detailed Balance

$$\pi(i)q(i,j)\alpha(i,j) = \pi(j)q(j,i)\alpha(j,i)$$

1. Initialise $x^{(0)}$.
2. For $i = 0$ to $N-1$
   - Sample $u \sim \mathcal{U}_{[0,1]}$.
   - Sample $x^\star \sim q(x^\star|x^{(i)})$.
   - If $u < \mathcal{A}(x^{(i)}, x^\star) = \min\left\{1, \frac{p(x^\star)q(x^{(i)}|x^\star)}{p(x^{(i)})q(x^\star|x^{(i)})}\right\}$

     $$x^{(i+1)} = x^\star$$

     else

     $$x^{(i+1)} = x^{(i)}$$

$$\alpha(i,j) = \min\{1, \frac{\pi(j)q(j,i)}{\pi(i)q(i,j)}\}$$

Special Cases of MHA

- Independent Sampler: Assuming that the proposal is independent of the current state

$$q(x^{(j)}|x^{(i)}) = q(x^{(j)})$$

$$\alpha(i,j) = \min\left\{1, \frac{p(x^{(j)})q(x^{(i)})}{p(x^{(i)})q(x^{(j)})}\right\} = \min\left\{1, \frac{w(x^{(j)})}{w(x^{(i)})}\right\}$$

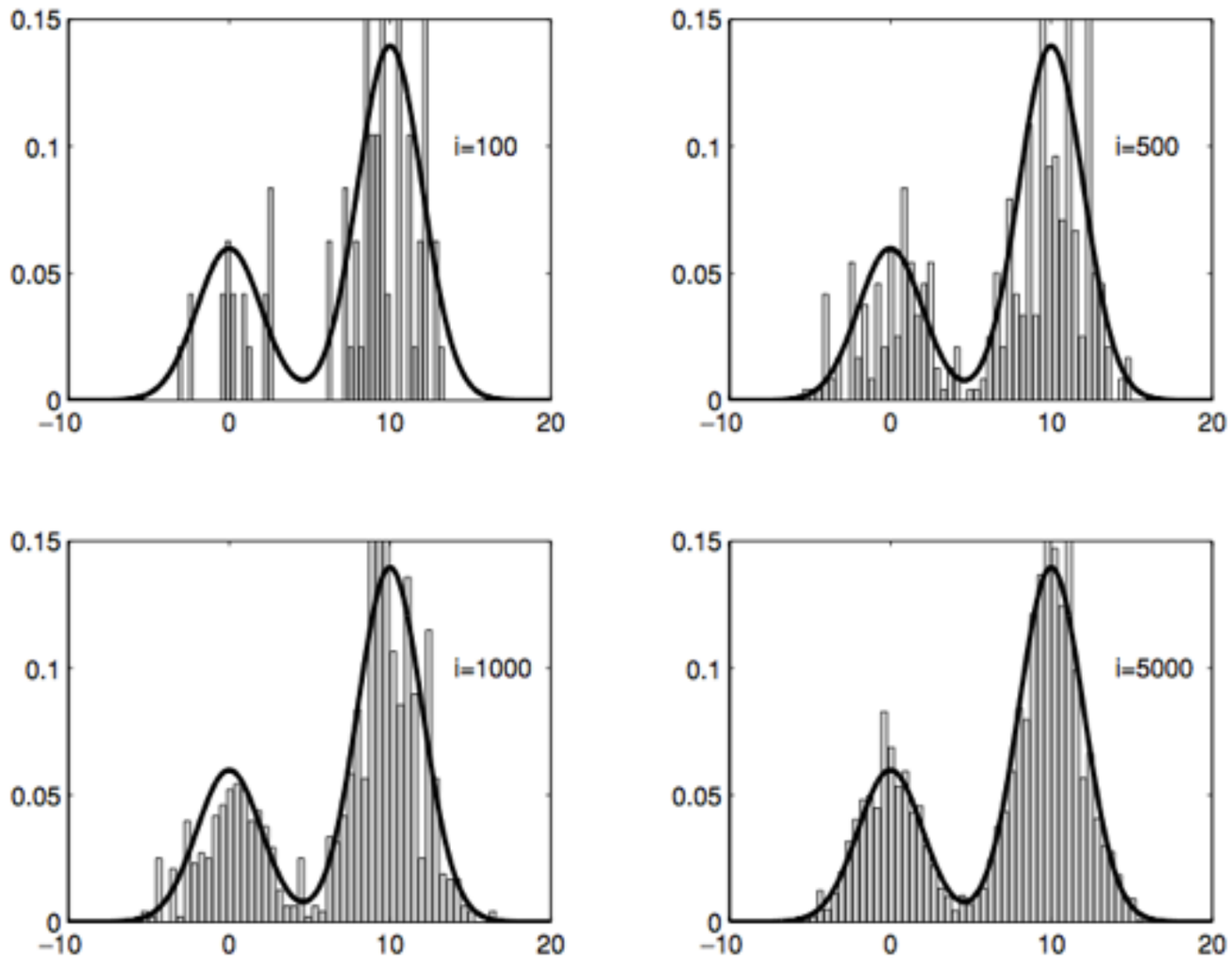
- Metropolis Sampler: Assuming that the proposal is symmetric

$$q(x^{(j)}|x^{(i)}) = q(x^{(i)}|x^{(j)})$$

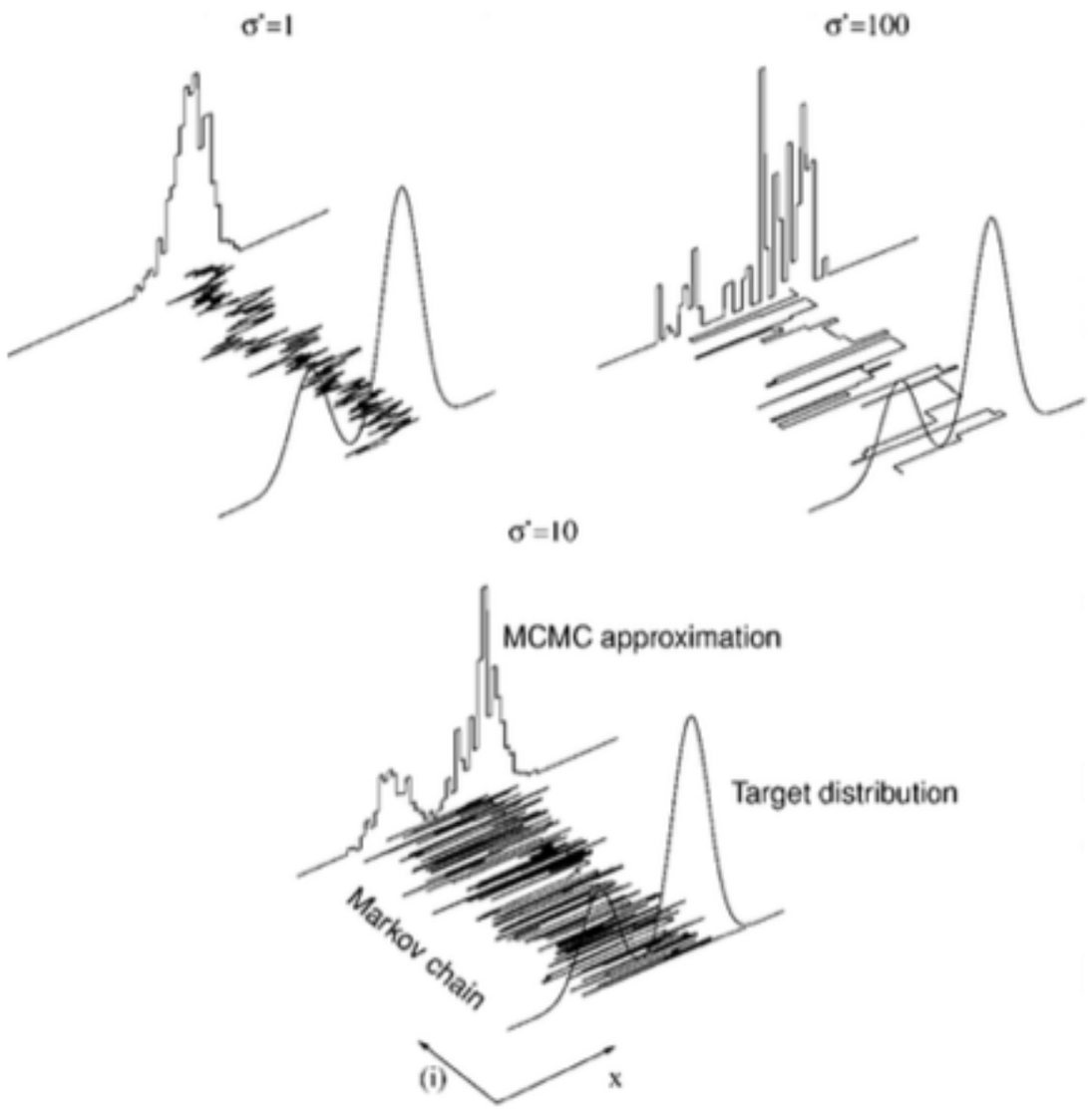
$$\alpha(i,j) = \min\left\{1, \frac{p(x^{(j)})}{p(x^{(i)})}\right\}$$

- Gibbs Sampler: Good property: no rejections

MHA Running Examples



MHA Running Examples



Simulated Annealing (SA)

- Annealing is the process of heating a solid until thermal stresses are released. Then, in cooling it very slowly to the ambient temperature until perfect crystals emerge. The quality of the results strongly depends on the cooling temperature. The final state can be interpreted as an energy state (crystaline potential energy) which is lowest if a perfectly crystal emerged

$$\Pr(\boldsymbol{x}) = \frac{1}{Z} \exp(-E(\boldsymbol{x})/k_B T)$$

$$\Pr(\boldsymbol{x} \to \boldsymbol{x}_p) = \begin{cases} 1, & \Delta E < 0; \\ \exp(-\Delta E/k_B T), & \Delta E \geq 0. \end{cases}$$

Simulated Annealing Algorithm

1. Initialize $\boldsymbol{x}^{(0)}$ and set $T_0, k = 0$.
2. For $i = 0$ to $N - 1$
    - Loop until converge
        - Sample $u \sim \mathcal{U}_{[0,1]}$.
        - Generate candidate state x^* according to current state $\boldsymbol{x}^{(k)}$
        - $\Delta E = E(\boldsymbol{x}^*) - E(\boldsymbol{x}^{(\boldsymbol{k})})$
        - Accept state $\boldsymbol{x}^*$ with probability:

$$\Pr(\boldsymbol{x}^{(k)} \to \boldsymbol{x}^*) = \begin{cases} 1, & \Delta E < 0; \\ \exp(-\Delta E / k_B T_i), & \Delta E \geq 0. \end{cases}$$

        if $u < \Pr(\boldsymbol{x} \to \boldsymbol{x}^*)$, set $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^*$, else set $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(\boldsymbol{k})}$
    - Set T_{i+1} according to cooling schedule.

Simulated Annealing Algorithm

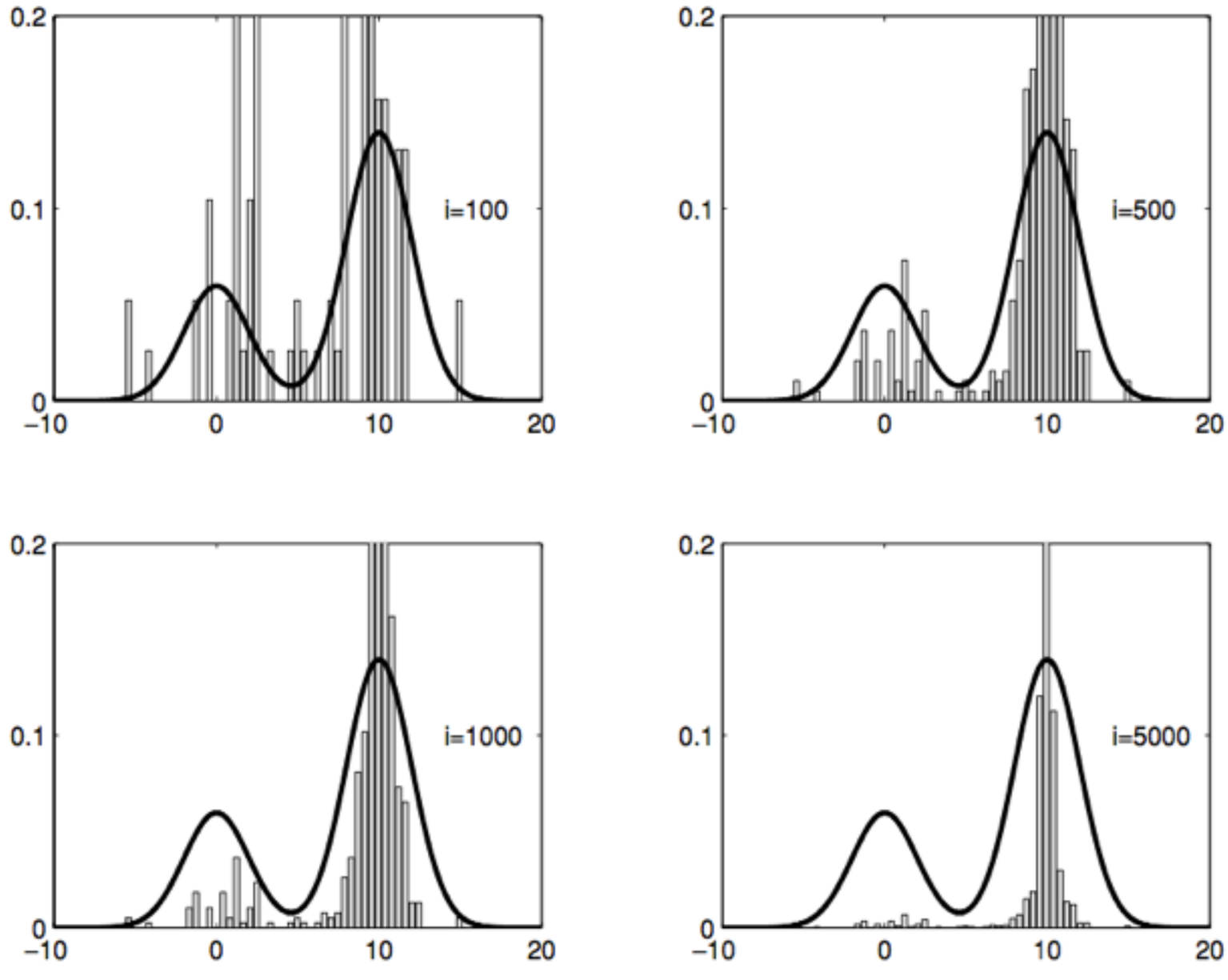
1. Initialise $x^{(0)}$ and set $T_0 = 1$.
2. For $i = 0$ to $N - 1$
   - Sample $u \sim \mathcal{U}_{[0,1]}$.
   - Sample $x^\star \sim q(x^\star | x^{(i)})$.
   - If $u < \mathcal{A}(x^{(i)}, x^\star) = \min\left\{1, \frac{p^{\frac{1}{T_i}}(x^\star) q(x^{(i)}|x^\star)}{p^{\frac{1}{T_i}}(x^{(i)}) q(x^\star|x^{(i)})}\right\}$

     $$x^{(i+1)} = x^\star$$

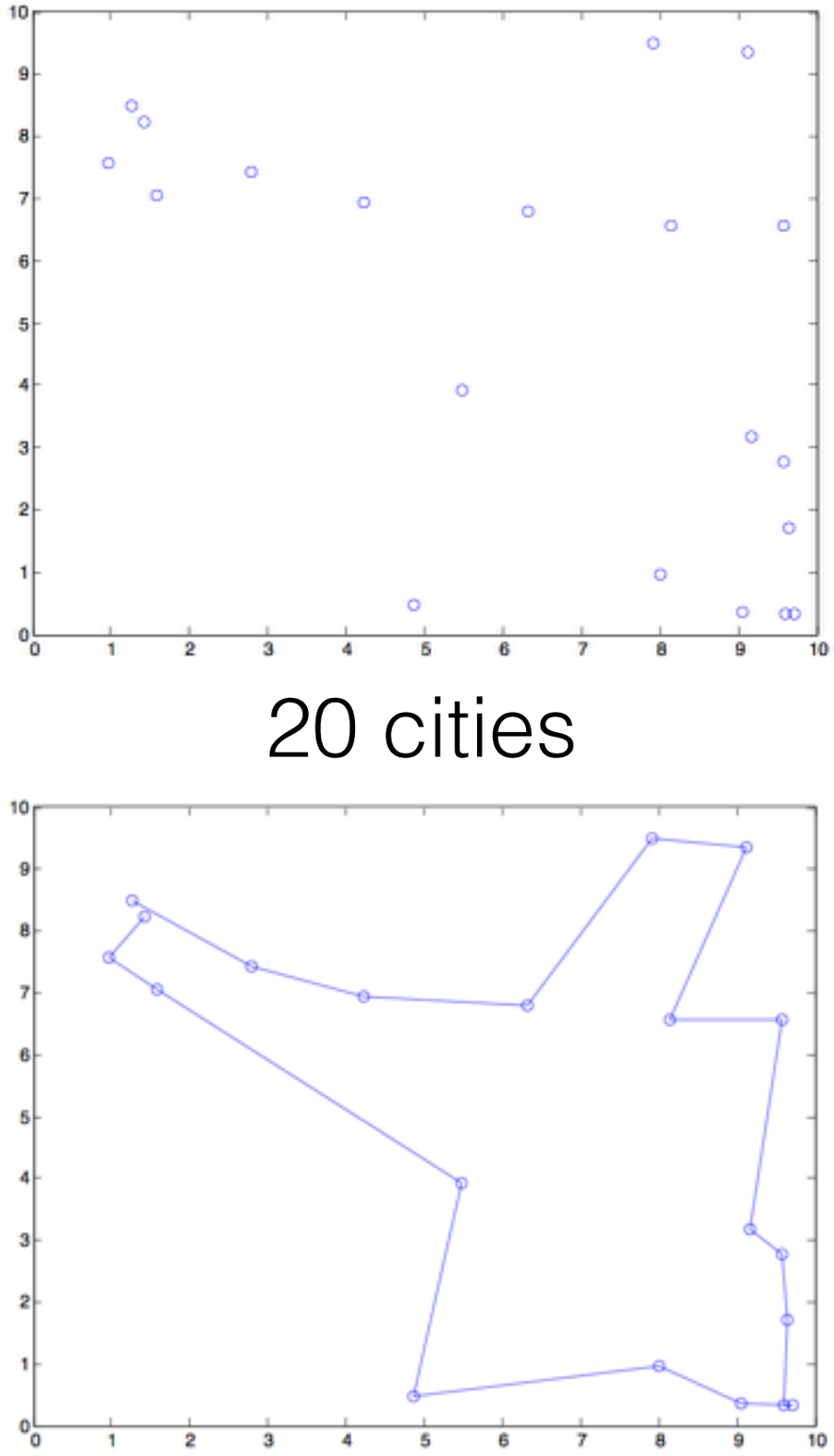
     else

     $$x^{(i+1)} = x^{(i)}$$
   - Set T_{i+1} according to a chosen cooling schedule.

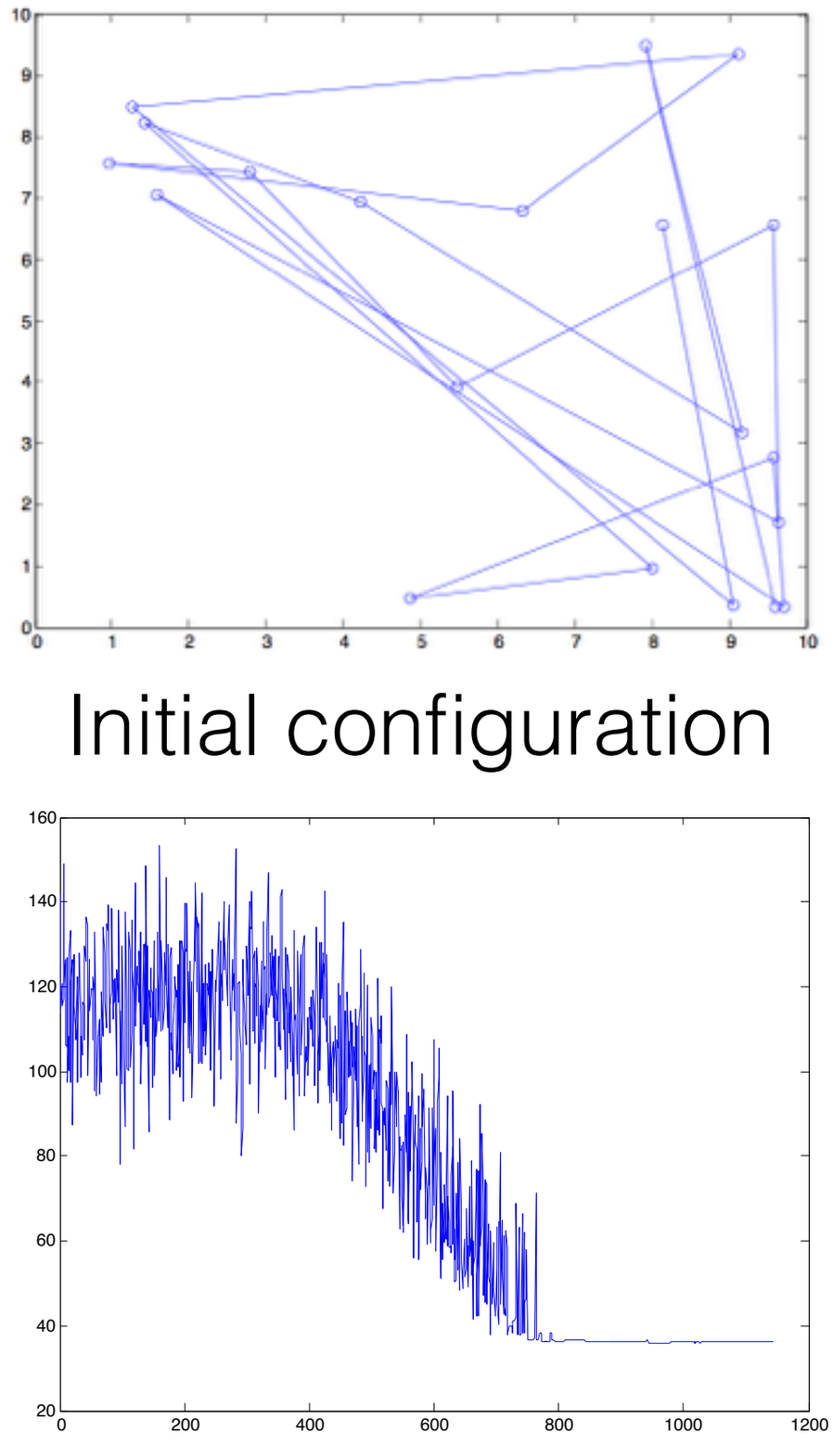
Toy Example: Maximization



Solve Traveling Salesman Problem with SA

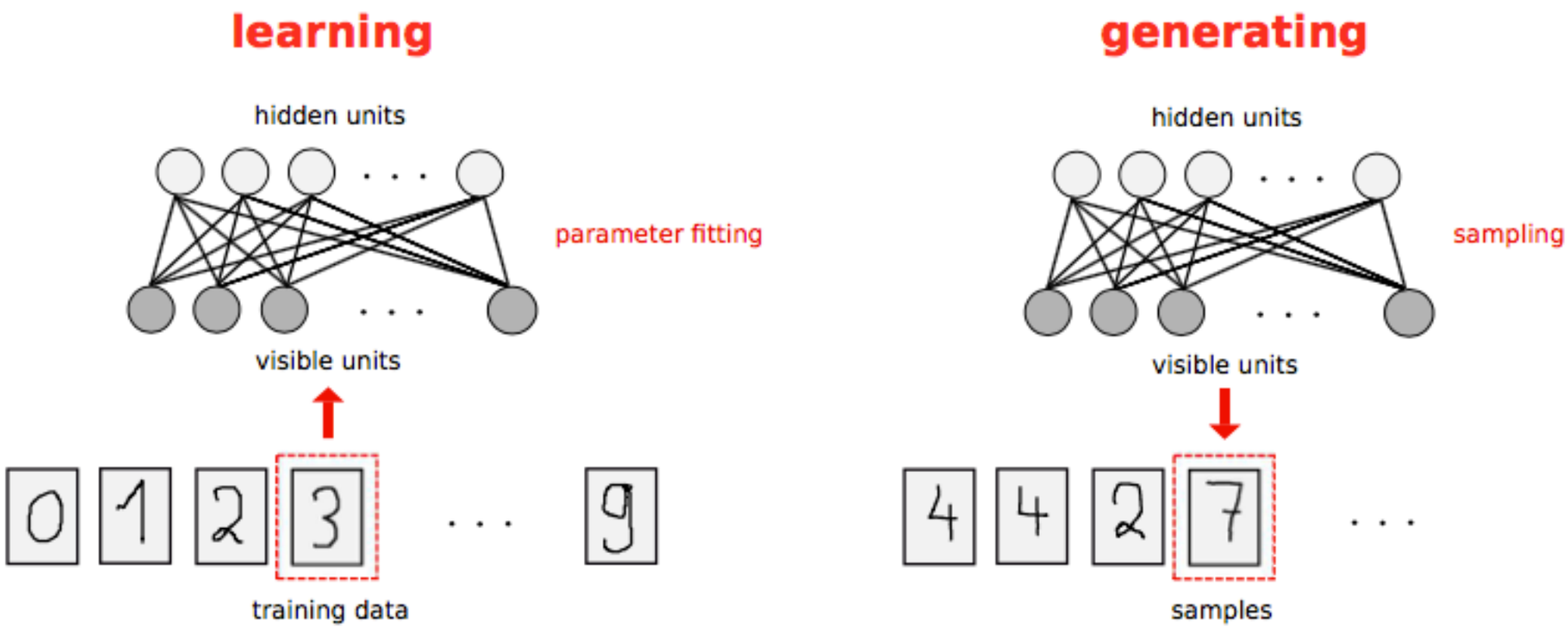


20 cities

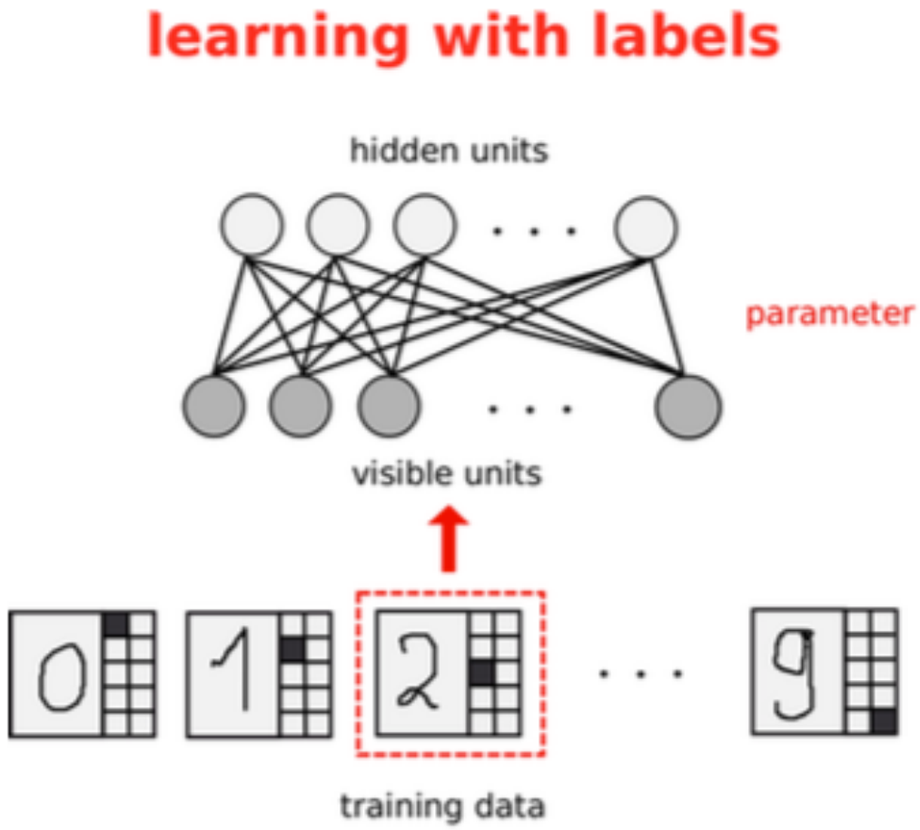


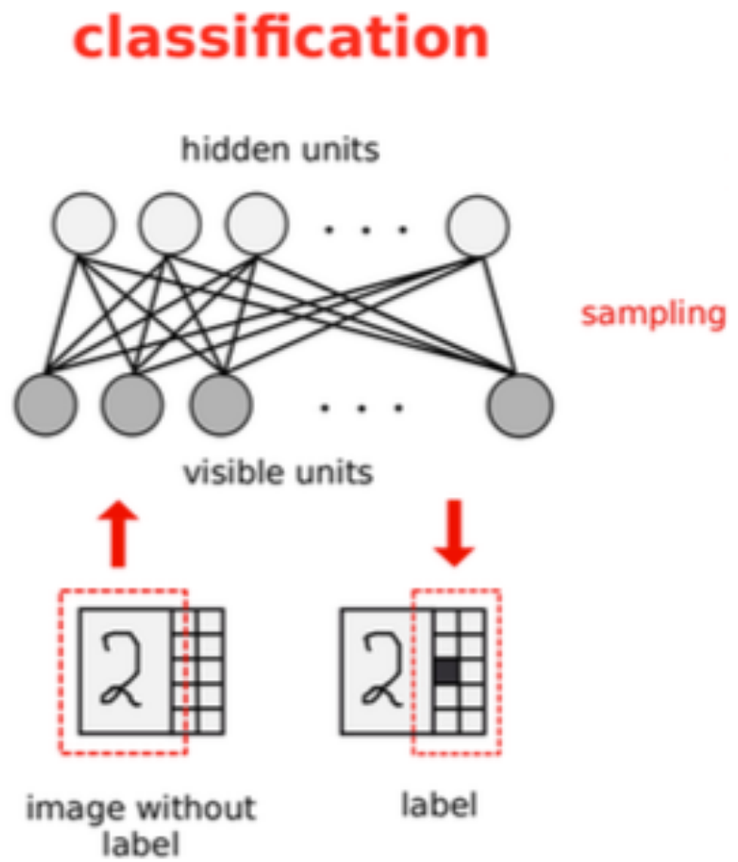
Initial configuration

Restricted Boltzmann Machine (RBM)

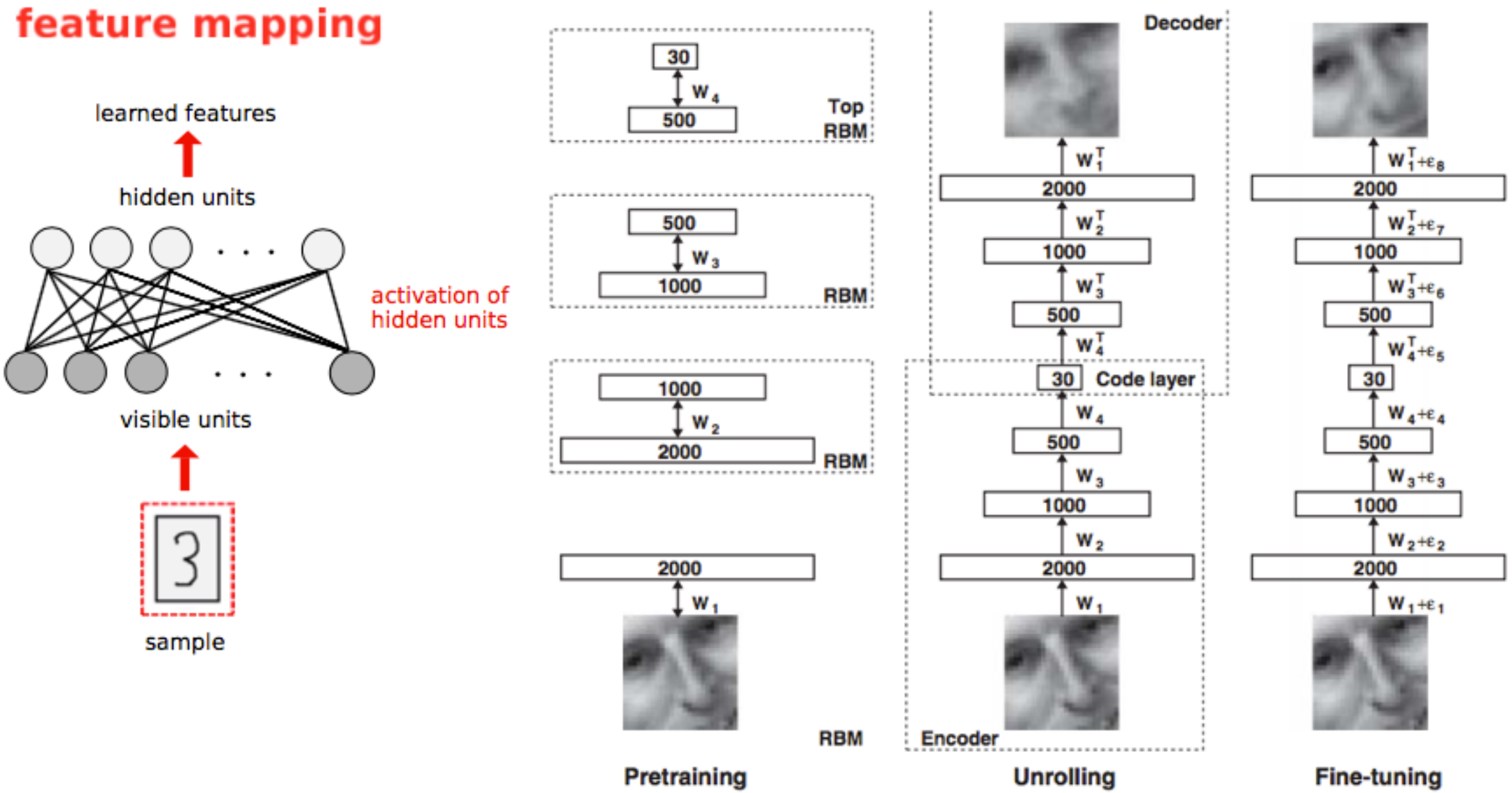


Restricted Boltzmann Machine (RBM)

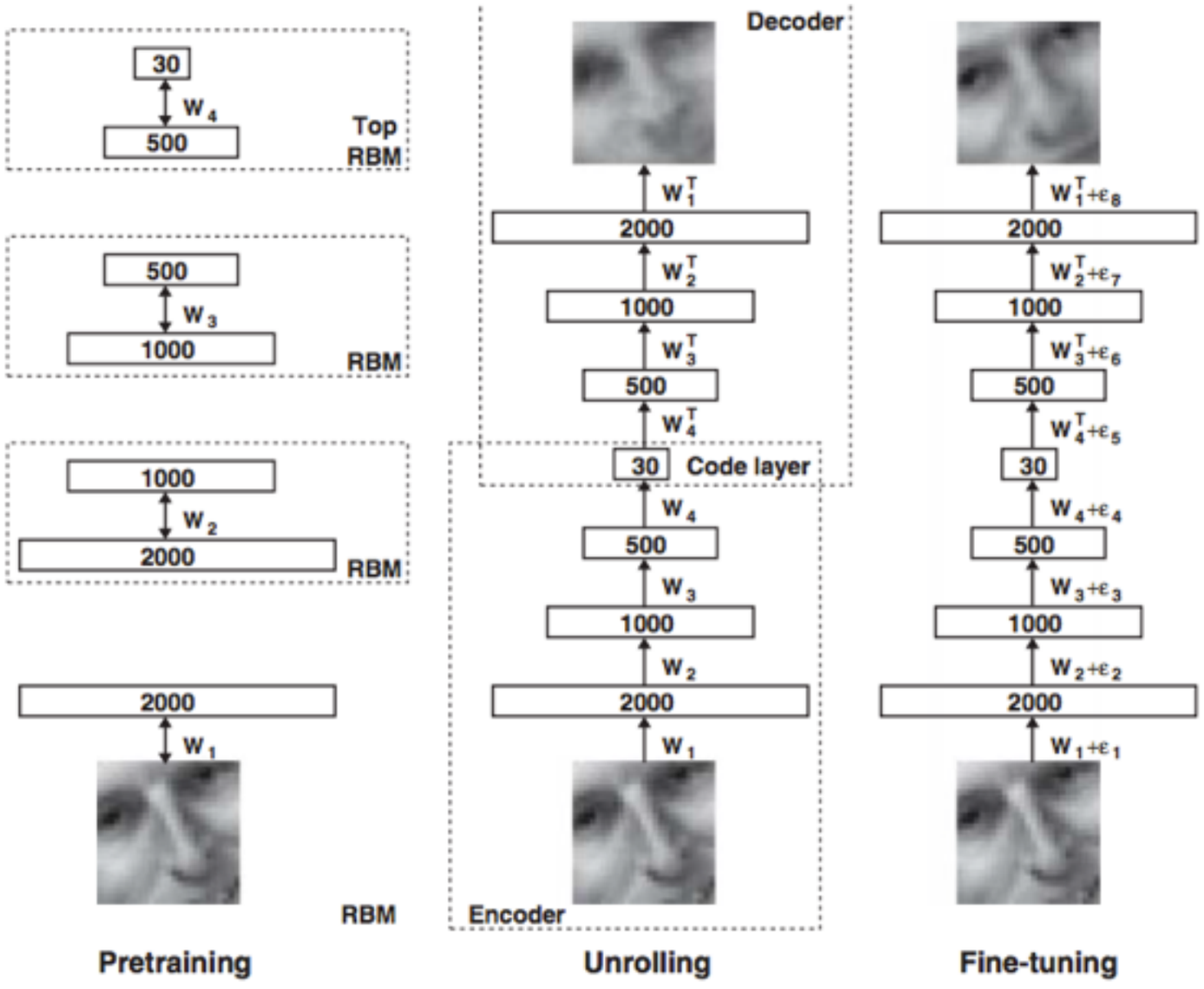


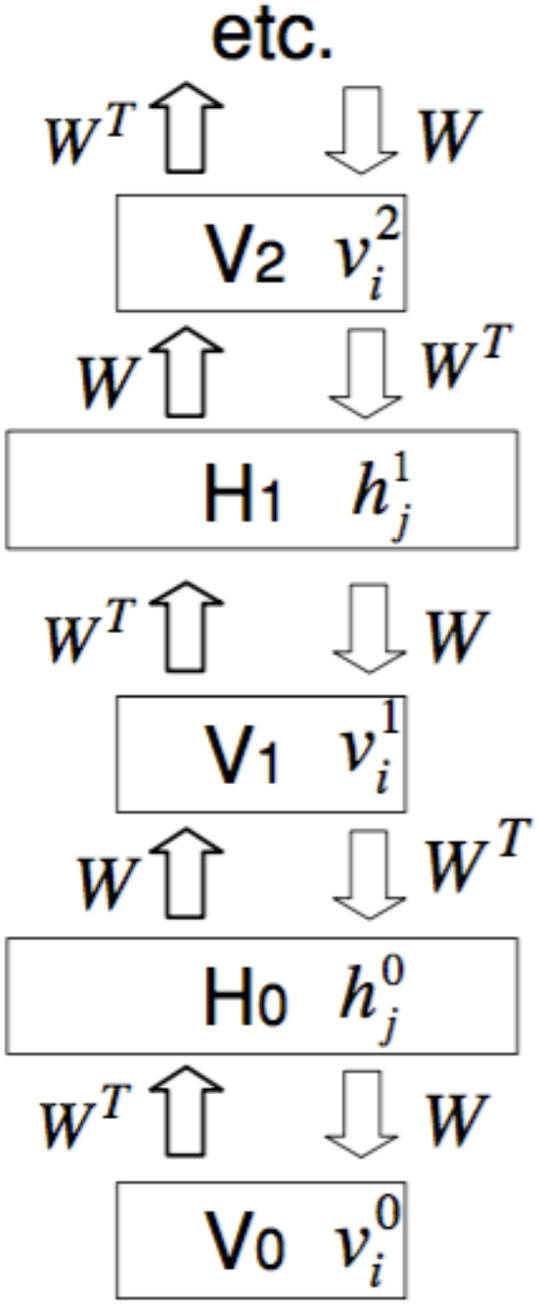


Restricted Boltzmann Machine (RBM)



Restricted Boltzmann Machine (RBM)





Basics to Learn RBM

- Fitting p(v) to data

$$p(\boldsymbol{v}) = \frac{1}{Z} \sum_{\boldsymbol{h}} \exp(-E(\boldsymbol{v}, \boldsymbol{h}))$$

- Notations

$$\begin{aligned} E(\boldsymbol{v}, \boldsymbol{h}) &= -\boldsymbol{v}^{\mathrm{T}} \boldsymbol{W} \boldsymbol{h} - \boldsymbol{b}^{\mathrm{T}} \boldsymbol{v} - \boldsymbol{c}^{\mathrm{T}} \boldsymbol{h} \\ &= -\sum_{i=1}^{m} \sum_{j=1}^{n} w_{ij} h_i v_j - \sum_{j=1}^{n} b_j v_j - \sum_{i=1}^{m} c_i h_i \\ &= -\boldsymbol{W} \circ \boldsymbol{A} - \boldsymbol{b}^{\mathrm{T}} \boldsymbol{v} - \boldsymbol{c}^{\mathrm{T}} \boldsymbol{h} \\ &= -\boldsymbol{\theta}^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x}) \end{aligned}$$

$$\boldsymbol{\theta} = [\boldsymbol{W}(:); \boldsymbol{b}; \boldsymbol{c}]$$

$$\boldsymbol{\phi}(\boldsymbol{x}) = [\boldsymbol{A}(:); \boldsymbol{v}; \boldsymbol{h}]$$

$$\boldsymbol{W} = \begin{pmatrix} w_{11} & w_{12} & \cdots & w_{1n} \\ w_{21} & w_{22} & \cdots & w_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{m1} & w_{m2} & \cdots & w_{mn} \end{pmatrix} \quad \boldsymbol{A} = \begin{pmatrix} h_1 v_1 & h_1 v_2 & \cdots & h_1 v_n \\ h_2 v_1 & h_2 v_2 & \cdots & h_2 v_n \\ \vdots & \vdots & \ddots & \vdots \\ h_m v_1 & h_m v_2 & \cdots & h_m v_n \end{pmatrix}$$

Learning RBM: Maximum Likelihood Estimation

- Given Observed Samples

$$S = \{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_\ell\}$$

- We can get

$$\frac{1}{\ell} \sum_{\boldsymbol{v} \in S} \frac{\partial \log p(\boldsymbol{v}; \boldsymbol{\theta})}{\partial w_{ij}} = \mathbb{E}_{p(\boldsymbol{h}|\boldsymbol{v};\boldsymbol{\theta})q(\boldsymbol{v})}[h_i v_j] - \mathbb{E}_{p(\boldsymbol{x};\boldsymbol{\theta})}[h_i v_j]$$

Intractable

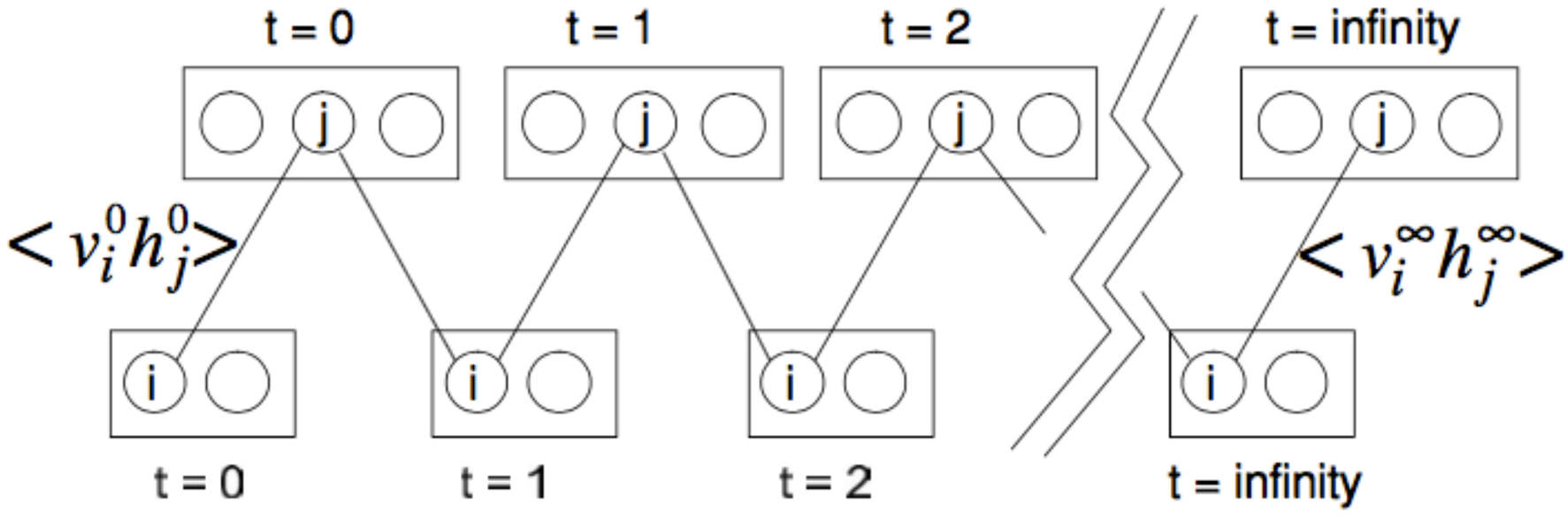
for

$$\frac{\partial \log p(\boldsymbol{v}_0; \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \mathbb{E}_{p(\boldsymbol{h}|\boldsymbol{v}_0;\boldsymbol{\theta})} \boldsymbol{\phi}(\boldsymbol{v}_0, \boldsymbol{h}) - \mathbb{E}_{p(\boldsymbol{x};\boldsymbol{\theta})}[\boldsymbol{\phi}(\boldsymbol{x})]$$

$$\frac{\partial \log p(\boldsymbol{v}_0; \boldsymbol{\theta})}{\partial w_{ij}} = \mathbb{E}_{p(\boldsymbol{h}|\boldsymbol{v}_0;\boldsymbol{\theta})}[h_i v_j] - \mathbb{E}_{p(\boldsymbol{x};\boldsymbol{\theta})}[h_i v_j]$$

Contrastive Divergence

$$\frac{\partial \log p(\boldsymbol{v}_0;\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \mathbb{E}_{p(\boldsymbol{h}|\boldsymbol{v}_0;\boldsymbol{\theta})}\boldsymbol{\phi}(\boldsymbol{v}_0,\boldsymbol{h}) - \mathbb{E}_{p(\boldsymbol{x};\boldsymbol{\theta})}[\boldsymbol{\phi}(\boldsymbol{x})]$$



$$\frac{\partial \log p(\boldsymbol{v}^{(0)};\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \approx \mathbb{E}_{p(\boldsymbol{h}|\boldsymbol{v}^{(0)})}[\boldsymbol{\phi}(\boldsymbol{v}^{(0)},\boldsymbol{h})] - \frac{1}{M}\sum_{i=1}^{M}\mathbb{E}_{p(\boldsymbol{h}|\boldsymbol{v}^{(k+i)})}[\boldsymbol{\phi}(\boldsymbol{v}^{(k+i)},\boldsymbol{h})]$$

$$\mathrm{CD}_k(\boldsymbol{\theta},\boldsymbol{v}^{(0)}) = \mathbb{E}_{p(\boldsymbol{h}|\boldsymbol{v}^{(0)})}[\boldsymbol{\phi}(\boldsymbol{v}^{(0)},\boldsymbol{h})] - \mathbb{E}_{p(\boldsymbol{h}|\boldsymbol{v}^{(k)})}[\boldsymbol{\phi}(\boldsymbol{v}^{(k)},\boldsymbol{h})]$$

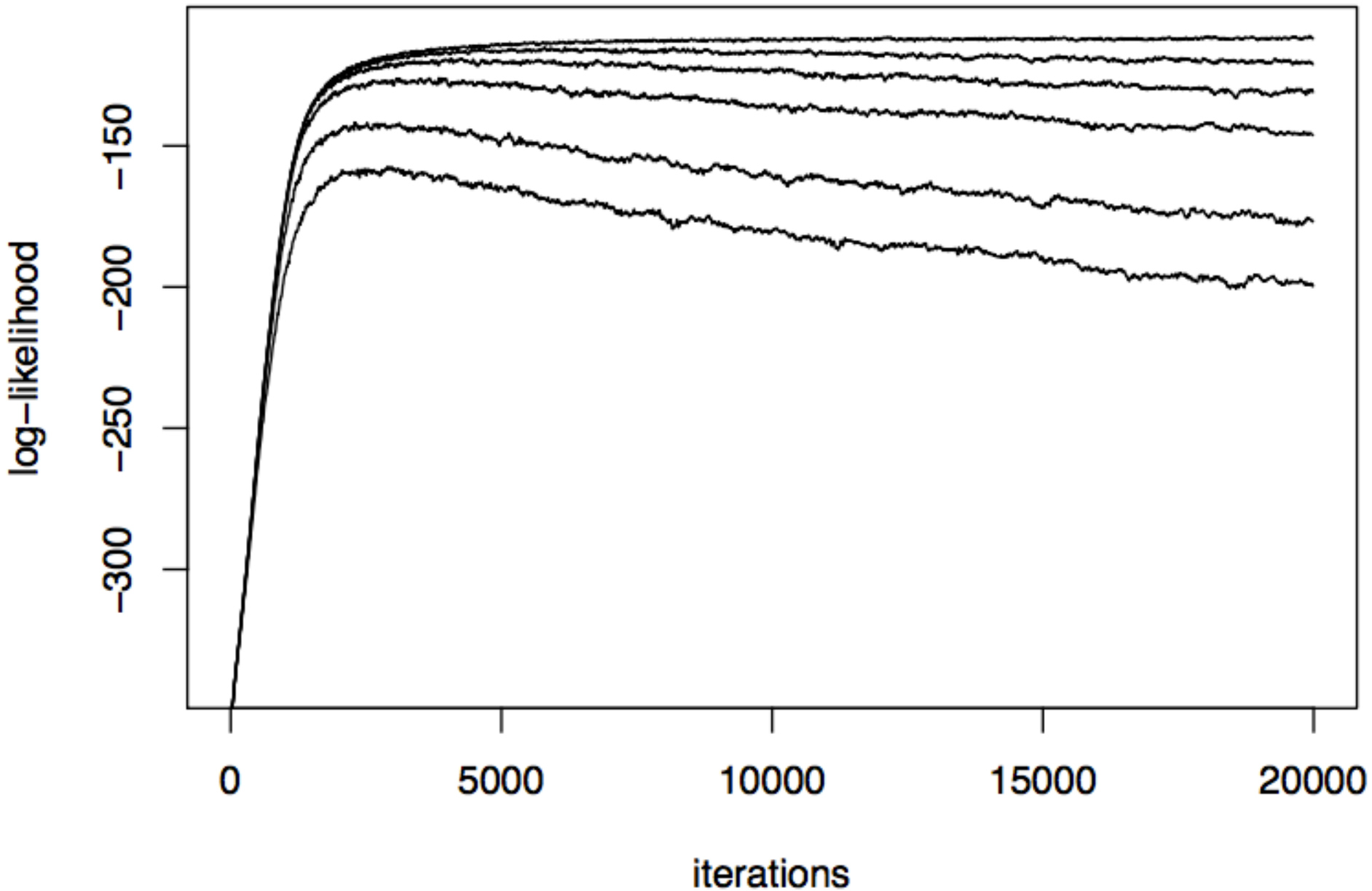
k-Step Contrastive Divergence Alg.

Algorithm 1: k-step contrastive divergence

Input: RBM $(V_1, \ldots, V_m, H_1, \ldots, H_n)$, training batch S
Output: gradient approximation Δw_{ij}, Δb_j and Δc_i for $i = 1, \ldots, n$, $j = 1, \ldots, m$

init $\Delta w_{ij} = \Delta b_j = \Delta c_i = 0$ for $i = 1, \ldots, n$, $j = 1, \ldots, m$
forall the $\boldsymbol{v} \in S$ **do**
  $\boldsymbol{v}^{(0)} \leftarrow \boldsymbol{v}$
  for $t = 0, \ldots, k-1$ **do**
    for $i = 1, \ldots, n$ **do** sample $h_i^{(t)} \sim p(h_i \mid \boldsymbol{v}^{(t)})$
    ;
    for $j = 1, \ldots, m$ **do** sample $v_j^{(t+1)} \sim p(v_j \mid \boldsymbol{h}^{(t)})$
    ;
  for $i = 1, \ldots, n$, $j = 1, \ldots, m$ **do**
    $\Delta w_{ij} \leftarrow \Delta w_{ij} + p(H_i = 1 \mid \boldsymbol{v}^{(0)}) \cdot v_j^{(0)} - p(H_i = 1 \mid \boldsymbol{v}^{(k)}) \cdot v_j^{(k)}$
  for $j = 1, \ldots, m$ **do**
    $\Delta b_j \leftarrow \Delta b_j + v_j^{(0)} - v_j^{(k)}$
  for $i = 1, \ldots, n$ **do**
    $\Delta c_i \leftarrow \Delta c_i + p(H_i = 1 \mid \boldsymbol{v}^{(0)}) - p(H_i = 1 \mid \boldsymbol{v}^{(k)})$

CD-k Running Examples



Note: from bottom to top k = 1, 2, 5, 10, 20, 100

Other Algorithms for Training RBM

- Persistent Contrastive Divergence (PCD)
- Fast Persistent Contrastive Divergence (FPCD)
- Stochastic Approximation Procedure (SAP)
- Stochastic Approximation Procedure with Tempered Transitions (Trans-SAP)
- Parallel Tempering (PT)

Stochastic Approximation Procedure with Tempered Transitions

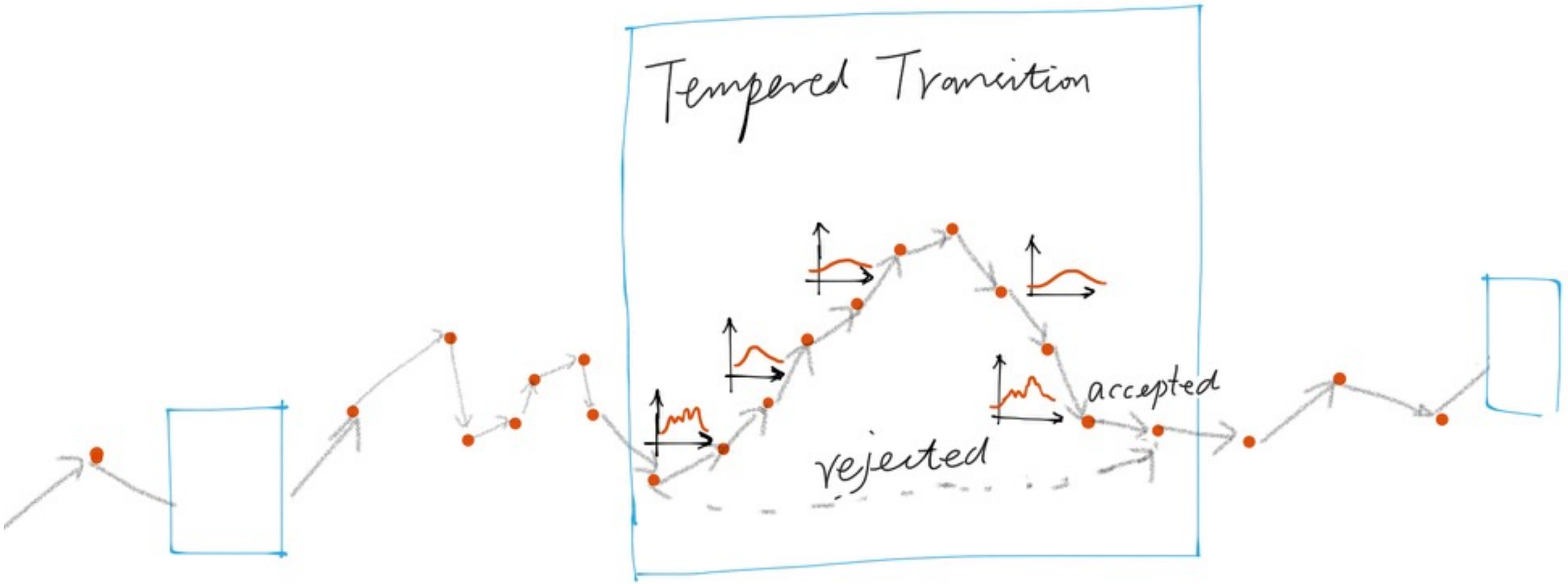
Algorithm 1 Stochastic Approximation Procedure.

Given an observation $\mathbf{x}_0$. Randomly initialize θ^1 and M sample particles $\{\mathbf{x}^{1,1}, ..., \mathbf{x}^{1,M}\}$.
for $t = 1 : T$ (number of iterations) **do**
  for $m = 1 : M$ (number of parallel Markov chains) **do**
    Sample $\mathbf{x}^{t+1,m}$ given $\mathbf{x}^{t,m}$ using transition operator $T_{\theta^t}(\mathbf{x}^{t+1,m} \leftarrow \mathbf{x}^{t,m})$.
  end for
  Update: $\theta^{t+1} = \theta^t + \alpha_t \left[\phi(\mathbf{x}_0) - \frac{1}{M}\sum_{m=1}^{M} \phi(\mathbf{x}^{t+1,m})\right]$.
  Decrease α_t.
end for

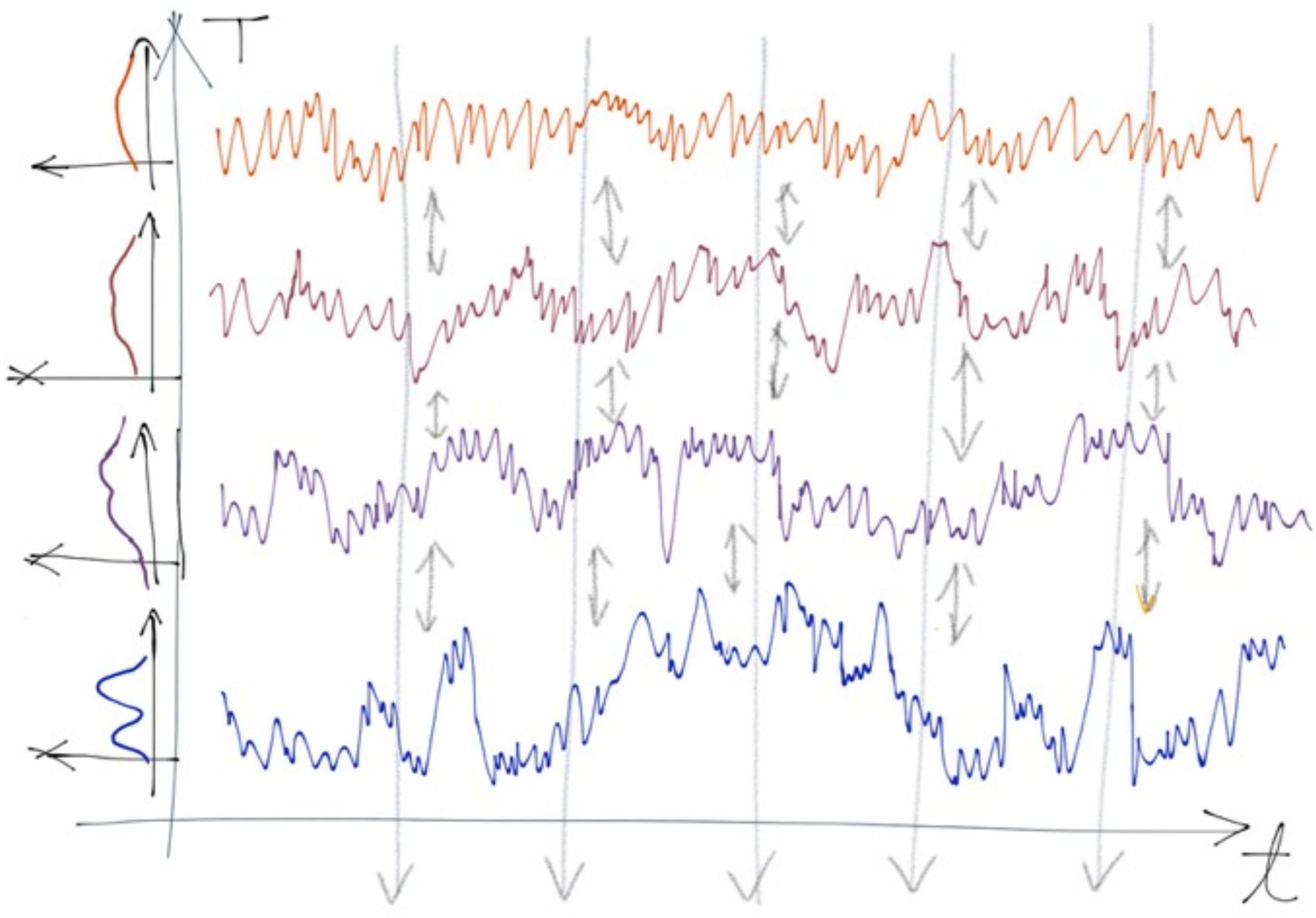
Algorithm 2 Tempered Transitions Run.

Initialize $\beta_0 < \beta_1 < ... < \beta_S = 1$. Given a current state $\mathbf{x}^S$.
for $s = S - 1 : 0$ (Forward pass) **do**
  Sample $\mathbf{x}^s$ given $\mathbf{x}^{s+1}$ using $T_s(\mathbf{x}^s \leftarrow \mathbf{x}^{s+1})$.
end for
Set $\tilde{\mathbf{x}}^0 = \mathbf{x}^0$.
for $s = 0 : S - 1$ (Backward pass) **do**
  Sample $\tilde{\mathbf{x}}^{s+1}$ given $\tilde{\mathbf{x}}^s$ using $\widetilde{T}_s(\tilde{\mathbf{x}}^{s+1} \leftarrow \tilde{\mathbf{x}}^s)$.
end for
Accept a new state $\tilde{\mathbf{x}}^S$ with probability: $\min\left[1, \prod_{s=1}^{S} p^*(\mathbf{x}_s)^{\beta_{s-1}-\beta_s}\, p^*(\tilde{\mathbf{x}}_s)^{\beta_s-\beta_{s-1}}\right]$.

Stochastic Approximation Procedure with Tempered Transitions



Parallel Tempering



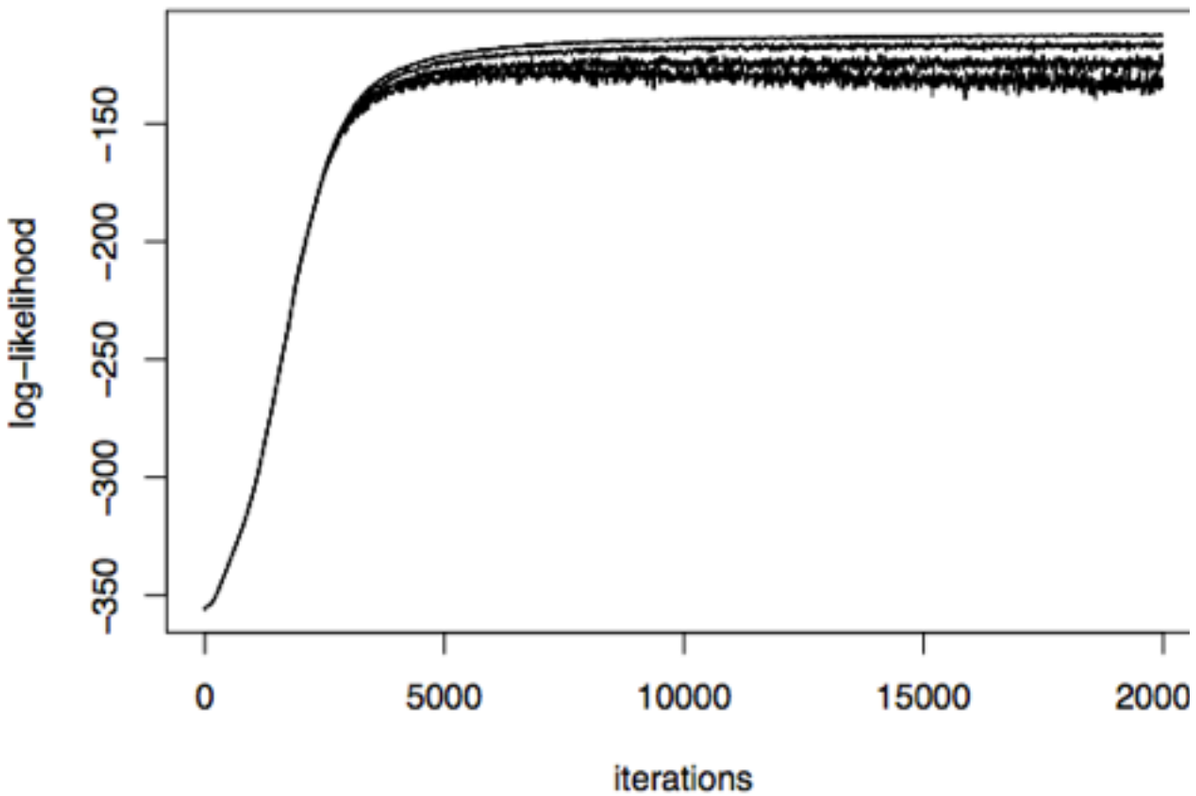
Parallel Tempering

1) Create a sequence of RBMs $(R_0, R_1, \cdots, R_K)$ such that parameters of R_k are $\boldsymbol{\Theta}_k = (T_k \mathbf{W}, \mathbf{b}, \mathbf{c})$, where $0 \leq T_0 < T_1 < \cdots < T_K = 1$.
2) Create an empty set of samples $\boldsymbol{X} = \{\}$.
3) Set $\mathbf{x}_0 = (\mathbf{x}_{0,0}, \cdots, \mathbf{x}_{K,0})$ such that every $\mathbf{x}_{k,0}$ is a uniformly distributed random vector (or use old ones from the previous epoch).
4) For $m = 1$ to M, repeat
   a) Sample $\mathbf{x}_m = (\mathbf{x}_{0,m}, \cdots, \mathbf{x}_{K,m})$ from the sequence of RBMs such that $\mathbf{x}_{k,m}$ is sampled by one-step Gibbs sampling starting from $\mathbf{x}_{k,m-1}$.
   b) For $j = K$ to 1, repeat
      - Swap $\mathbf{x}_{j,m}$ and $\mathbf{x}_{j-1,m}$ according to $P_{\text{swap}}(\mathbf{x}_{j,m}, \mathbf{x}_{j-1,m})$ computed using ■.
   c) Add $\mathbf{x}_{K,m}$ to $\boldsymbol{X}$.
5) $\boldsymbol{X}$ is the set of samples collected by parallel tempering sampling.

■ $$P_{\text{swap}}(\mathbf{x}_{T_1}, \mathbf{x}_{T_2}) = \min\left(1, \frac{P_{T_1}(\mathbf{x}_{T_2}) P_{T_2}(\mathbf{x}_{T_1})}{P_{T_1}(\mathbf{x}_{T_1}) P_{T_2}(\mathbf{x}_{T_2})}\right),$$

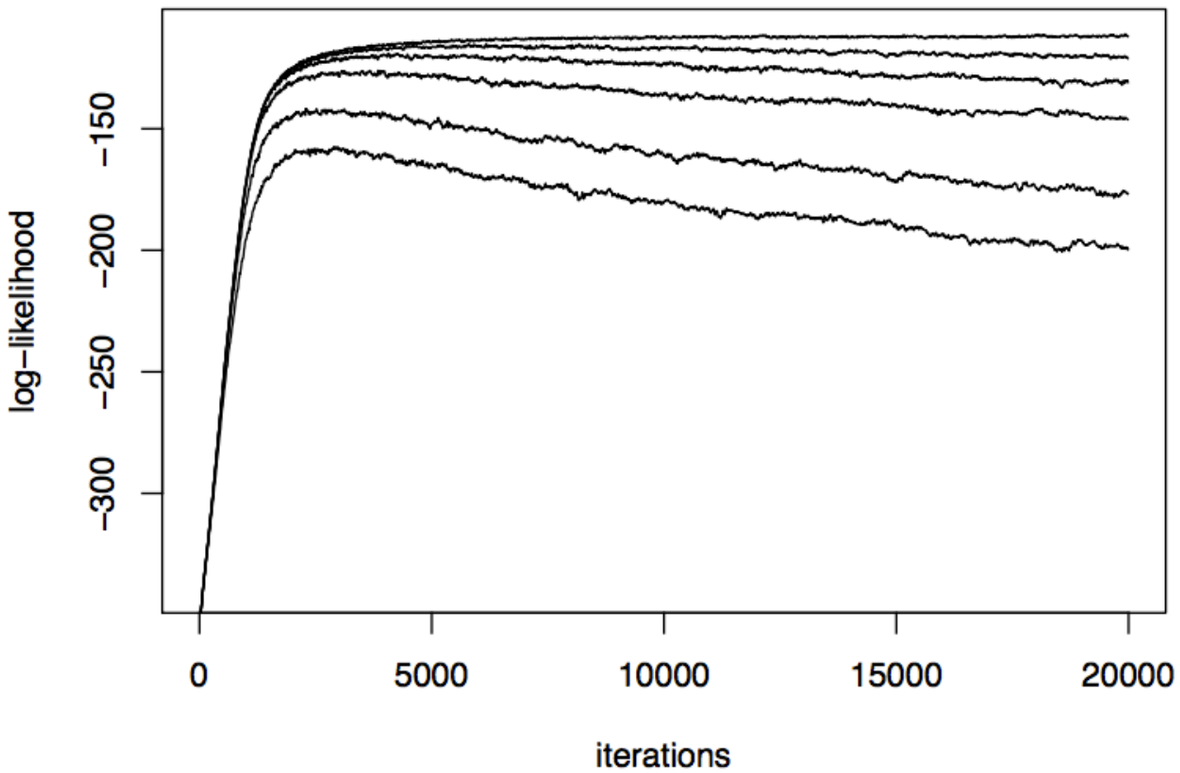
Parallel Tempering v.s. CD-k: Running Examples (1/2)

PT



Note: bottom to top M = 4,5,10,50

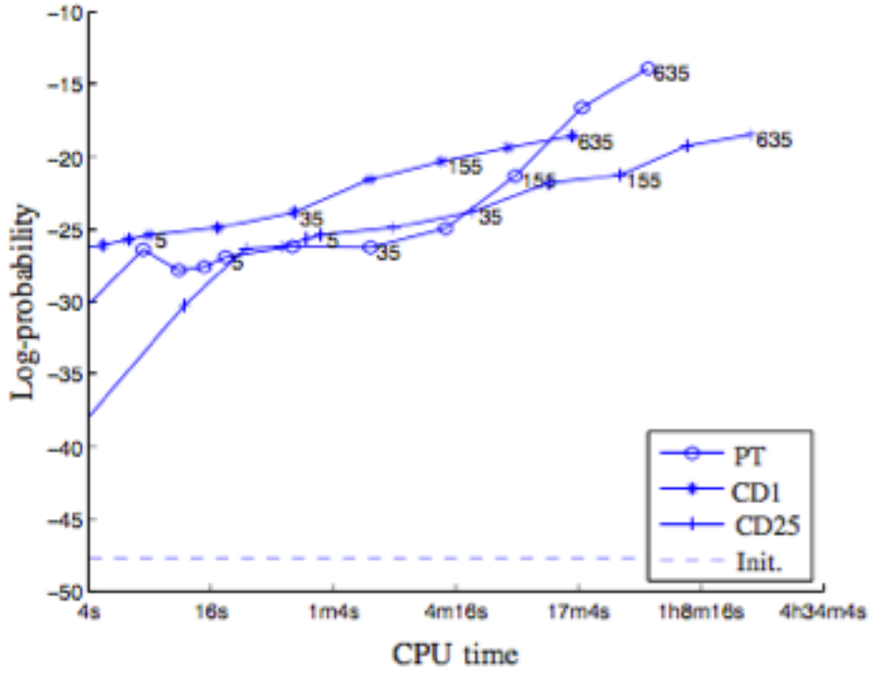
CD-k

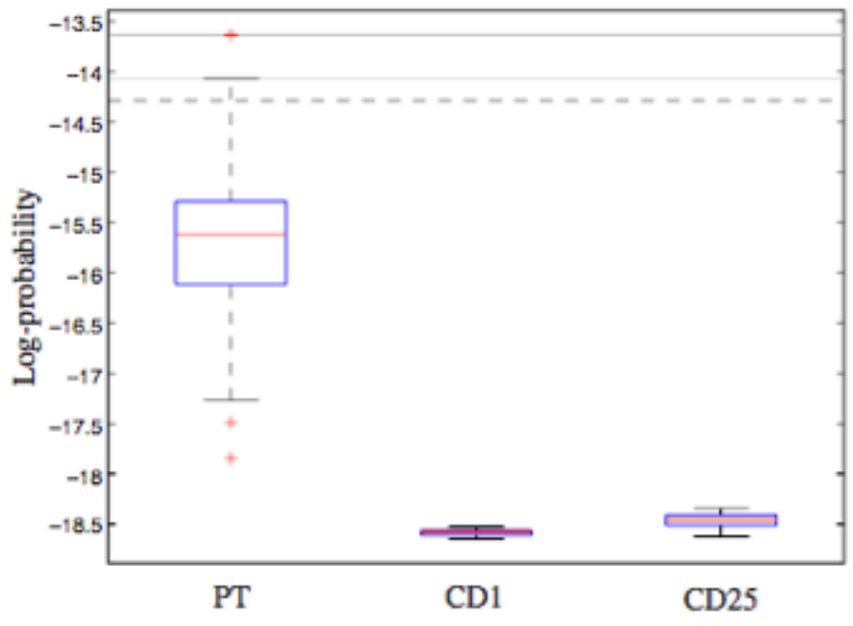


Note: from bottom to top k = 1, 2, 5, 10, 20, 100

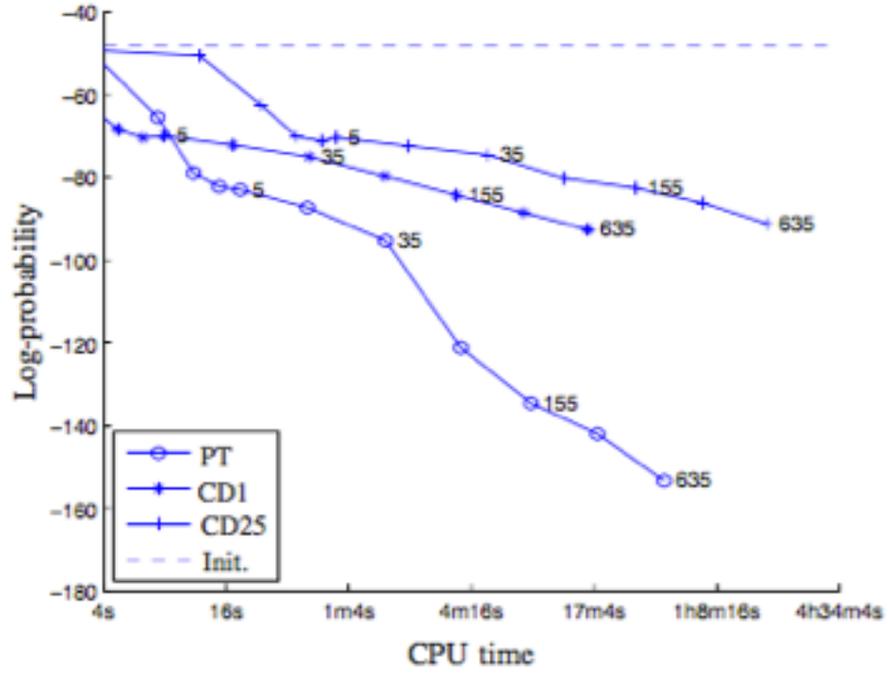
Parallel Tempering v.s. CD-k: Running Examples (2/2)

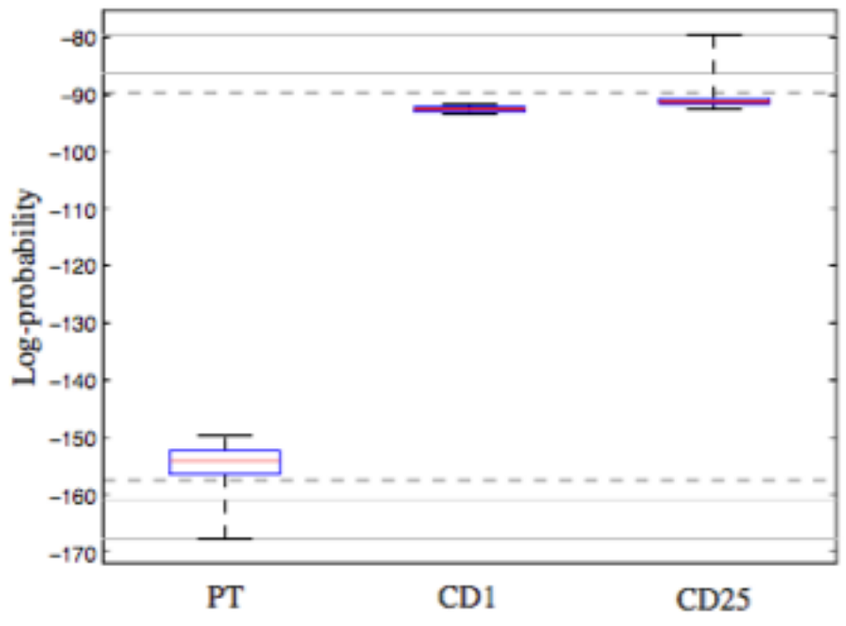
Positive Samples





Negative Samples





References

- Andrieu, Christophe, Nando de Freitas, Arnaud Doucet, and Michael I Jordan. ‘An Introduction to MCMC for Machine Learning.’, *Machine learning*, 2003

- Fischer, Asja, and Christian Igel. ‘Training restricted Boltzmann machines: An introduction.’, *Pattern Recognition*, 2014

- Salakhutdinov, Ruslan. ‘Learning in Markov Random Fields using Tempered Transitions.’, NIPS, 2009

- Geoffrey E. Hinton, Simon Osindero, Simon Osindero, and Yee-Whye Teh. ‘A Fast Learning Algorithm for Deep Belief Nets’, *Neural computation*, 2006

- Desjardins, Guillaume, Aaron C Courville, Yoshua Bengio, Pascal Vincent, and Olivier Delalleau. ‘Parallel Tempering for Training of Restricted Boltzmann Machines ’, *AISTATS*, 2010

- Neal, Radford M. ‘Sampling from multimodal distributions using tempered transitions’, Statistics and computing, 1996

- Geoffrey E. Hinton. ‘Reducing the Dimensionality of Data with Neural Networks’, Science, 2006

- Cho, Kyunghyun, Tapani Raiko, and Alexander Ilin. ‘Parallel tempering is efficient for learning restricted Boltzmann machines’, International Joint Conference on Neural Networks (IJCNN), 2010

Thanks