InfoLab

# Chapter 16~17. Structured Probabilistic Models for Deep Learning & Monte Carlo Methods


2018-01-24

Jinwook Kim

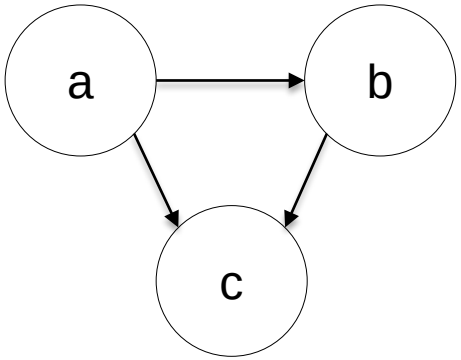
DGIST 대구경북과학기술원
Daegu Gyeongbuk Institute of Science & Technology

# Contents

# Structured Probabilistic Models

- **Graphical expressions of the (conditional) dependence structure between random variables (RVs)**
  - a.k.a. graphical models

- **Graphs for describing random variables and their structures**
  - vertices: states of random variables
  - edges: relationships of random variables




$$p(a,b,c) = p(a)p(b|a)p(c|a,b)$$

*$p(\cdot)$: probability distribution
*$a, b, c$: random variables

# Structured Modeling for Deep Learning

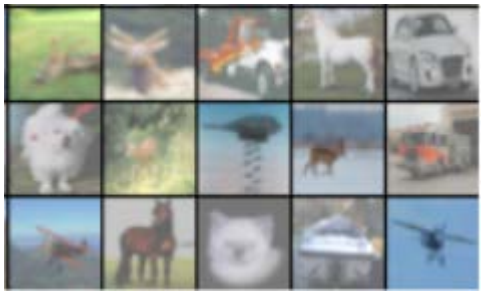
- **The goal of deep learning is to scale machine learning to solve real world problems (AI)**

- **Most require a complete understanding of the entire structure of the input**
  - density estimation
  - denoising
  - missing value imputation
  - sampling

original image | noised image | denoising output

A denoising example.

original training images



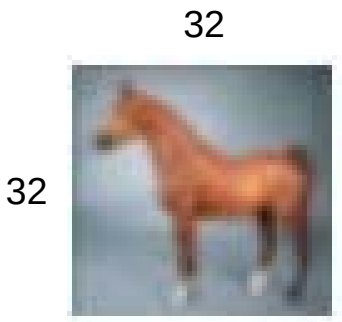
sampled images

A sampling example.

# The Challenge of Unstructured Modeling

- **Naïve approach of representing a distribution over a random vector $\boldsymbol{x}$, $\boldsymbol{P(x)}$**
  - $\boldsymbol{x}$ contains $n$ discrete variables
  - each variable takes on $k$ values
  - $P(\boldsymbol{x})$ requires to lookup $k^n$ space

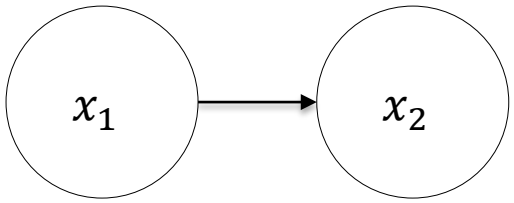
- **Example: modeling CIFAR-10**
  - 32x32 RGB color images ($n = 3072$ pixels)
  - let, each pixel has a 0/1 binary value ($k = 2$)
  - # possible binary images: $2^{3072}$ $(> 10^{800})$



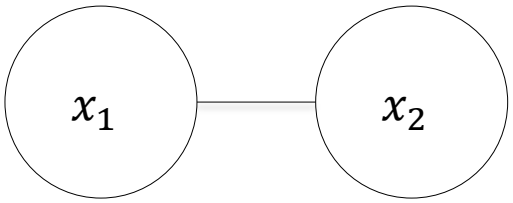
# Using Graphs to Describe Model Structure

- **Formal framework for modeling only direct interactions between RVs**
  - fewer parameters
  - reduced computational cost

- **Two categories of graphical models:**
  - directed acyclic graphs
  - undirected graphs



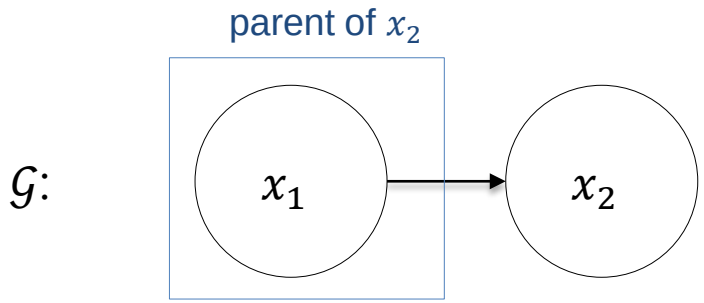

directed acyclic graph




undirected graph

# Directed Models

- **Describe conditional probability distribution over one RV via another**



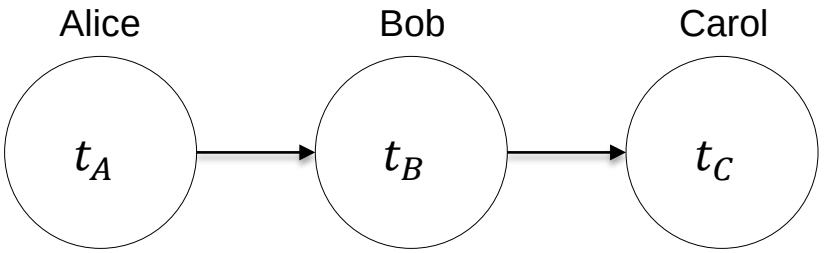

- **Defined by a set of local conditional probability distributions**

$$p(\boldsymbol{x}) = \prod_{\boldsymbol{i}} p(x_i | Pa_{\mathcal{G}}(x_i))$$

  - $\mathcal{G}$: a directed acyclic graph
  - $Pa_{\mathcal{G}}(x_i)$: parents of $x_i$ in $\mathcal{G}$

# Relay Race Example

- **Modeling the finish times of a team in a relay race**
  - three runners: Alice, Bob, and Carol
  - $t_i$: finishing times of each runner
    (assume, each $t_i$ can have 100 possible values)




- **Probability distributions of relay race**

$$p(t_A, t_B, t_C) = p(t_A)p(t_B|t_A)p(t_C|t_B)$$

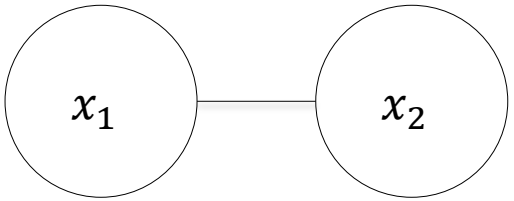
  - lookup space for graphical model: $100 + 100^2 + 100^2 = 20{,}100$

(*c.f.*, naïve & unstructured approach: $100^3 = 1{,}000{,}000$)

# Undirected Models

- **Describe probability distribution of RVs which have no intrinsic direction or operate both directions**
  - ➢ there is no conditional probability distribution between RVs

$\mathcal{G}$:



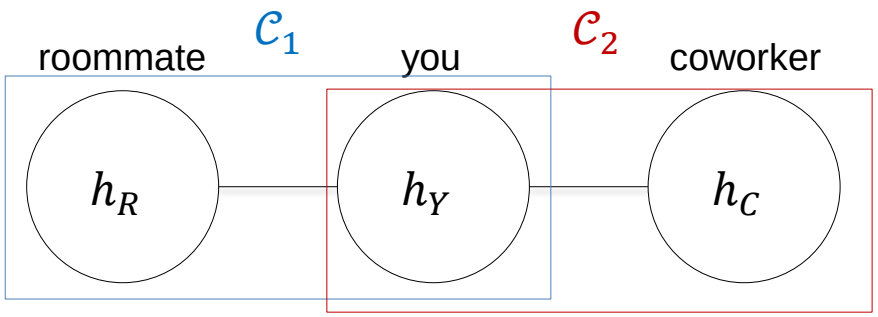

- **Defined by an *unnormalized* probability distribution**

$$\tilde{p}(\boldsymbol{x}) = \prod_{\mathcal{C} \in \mathcal{G}} \phi(\mathcal{C})$$

  - ➢ a factor $\phi(\mathcal{C})$ measures the affinity of the variables in $\mathcal{C}$ (non-negative)
    - $\mathcal{G}$: an undirected graph
    - $\mathcal{C}$: a clique in $\mathcal{G}$
    - $\phi$: a factor (or clique potential)

# Health State Example

- **Modeling the states of health of you, roommate, and coworker**
  - $h_i$: a state of health (1: good health, 0: with a cold)




$\phi(h_R, h_Y)$

| $c_1$ | $h_Y = 0$ | $h_Y = 1$ |
|---|---|---|
| $h_R = 0$ | 0.4 | 0.1 |
| $h_R = 1$ | 0.1 | 5 |

$\phi(h_Y, h_C)$

| $c_2$ | $h_Y = 0$ | $h_Y = 1$ |
|---|---|---|
| $h_C = 0$ | 0.2 | 0.1 |
| $h_C = 1$ | 0.1 | 5 |

- **Unnormalized probability distributions of health state**

$$\tilde{p}(\boldsymbol{x}) = \phi(h_R, h_Y)\phi(h_Y, h_C)$$

# Partition Function

- **Unnormalized → normalized probability distribution:**

$$p(\boldsymbol{x}) = \frac{1}{Z}\tilde{p}(\boldsymbol{x})$$

  - $\tilde{p}(\boldsymbol{x}) > 0$, for all $\boldsymbol{x}$
  - $Z$ is the value that results in the $p(\boldsymbol{x})$ summing (or integrating) to 1

- **Partition function $Z$:**

$$Z = \int \tilde{p}(\boldsymbol{x}) d\boldsymbol{x}$$

  - however, in the context of deep learning, $Z$ is usually intractable
  - approximation methods for $Z$ will be introduced in Chapter 18

# Probabilities for Health State

- $Z =$

| $h_R$ | $h_Y$ | $h_C$ | $\tilde{p}(x)$ $\phi(C_1)\phi(C_2)$ |
|---|---|---|---|
| 0 | 0 | 0 | 0.08 |
| 0 | 0 | 1 | 0.04 |
| 0 | 1 | 0 | 0.01 |
| 0 | 1 | 1 | 0.5 |
| 1 | 0 | 0 | 0.02 |
| 1 | 0 | 1 | 0.01 |
| 1 | 1 | 0 | 0.5 |
| 1 | 1 | 1 | 25 |
| $Z$ | | | 26.16 |

($x$)

$\phi(h_R, h_Y)$

| $C_1$ | $h_Y = 0$ | $h_Y = 1$ |
|---|---|---|
| $h_R = 0$ | 0.4 | 0.1 |
| $h_R = 1$ | 0.1 | 5 |

$\phi(h_Y, h_C)$

| $C_2$ | $h_Y = 0$ | $h_Y = 1$ |
|---|---|---|
| $h_C = 0$ | 0.2 | 0.1 |
| $h_C = 1$ | 0.1 | 5 |

- **All three people are in good health**

$$p(1,1,1) = \frac{1}{26.16} \times 5 \times 5 = 0.9556$$

- **Roommate and you are coughing but coworker is fine**

$$p(0,0,1) = \frac{1}{26.16} \times 0.4 \times 0.1 = 0.0015$$

# Directed vs. Undirected Models

- **Directed models are defined directly in terms of probability distributions**

- **Undirected models are defined more loosely by $\phi$ functions**
  - $\forall \boldsymbol{x},\ \tilde{p}(\boldsymbol{x}) > 0$
  - $\phi$ need to be converted into probability distributions

# Sampling from Directed Models

- **The advantage of directed models is that a simple and efficient procedure of ancestral sampling**
  1. sort $x_i$'s in the graph into a topological ordering
     let, $j > i$, $x_i$ is a parent of $x_j$
  2. sample $x_i \sim P(x_i)$,
  3. sample $x_j \sim P(x_j | Pa_{\mathcal{G}}(x_j) = x_i)$, and so on …

- **Pros and cons:**
  + very fast and convenient
  - only applies to directed graphical models
  - does not support every conditional sampling operation
    (e.g., sampling from posterior distribution given observed variables)

# Sampling from Undirected Models

- **Sampling from undirected models requires resolving cyclical dependencies**
  - every variables interacts with every other variables
  - there is no clear beginning point for sampling process

- **The conceptually simple approach is Gibbs sampling**
  - for $n$-dimensional vector of RV $\boldsymbol{x}$,
  - we interactively visit $x_i$ and draw samples conditioned on all the other variables
  - by repeating, asymptotically this process converges to sampling from the correct distribution

# Energy-Based Models (EBM)

- **Undirected models depend on the assumption that**

$$\forall \boldsymbol{x}, \qquad \tilde{p}(\boldsymbol{x}) > 0$$

- **A convenient way to enforce this condition is to use a EBM**

$$\tilde{p}(\boldsymbol{x}) = \exp(-E(\boldsymbol{x}))$$

  - $E$: energy function
  - $\exp(\boldsymbol{x})$: $e^{\boldsymbol{x}}$ ($e \approx 2.71818$ ...)

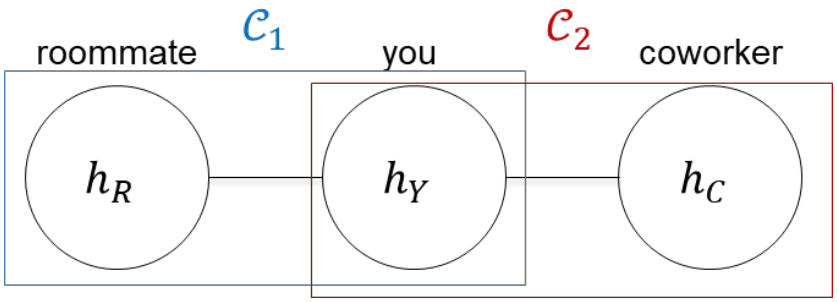
- **EBM captures dependencies by associating a scalar energy to each configuration of the variables**
  - ➢ energy function measures the goodness (or badness) of each possible configuration of variables

# EBM for Health State

$$\phi_E(\mathcal{C}_i) = \exp(-\phi(\mathcal{C}_i))$$

$$E(\mathcal{C}_1, \dots, \mathcal{C}_n) = \sum_i^n \phi_E(\mathcal{C}_i)$$

- $\mathcal{C}_i$: a clique in the health state graph
- $\phi$: a clique potential for health state
- $\phi_E$: a per-clique energy function
- $E$: an energy function




$$\begin{aligned}\tilde{p}(\boldsymbol{x}) &= \exp(E(\mathcal{C}_1, \mathcal{C}_2)) \\ &= \exp\Big(\big(\phi_E(\mathcal{C}_1) + \phi_E(\mathcal{C}_2)\big)\Big) \\ &= \exp\big(\exp\big(-\phi(\mathcal{C}_1)\big) + \exp\big(-\phi(\mathcal{C}_2)\big)\big) \\ &= \exp\Big(\exp\big(-\phi(\mathcal{C}_1)\big)\Big) \exp\Big(\exp\big(-\phi(\mathcal{C}_2)\big)\Big)\end{aligned}$$

# Probabilities for Health State on EBM

- $Z =$

| $h_R$ | $h_Y$ | $h_C$ | $\phi_E(C_1)$ | $\phi_E(C_2)$ | $\tilde{p}(\boldsymbol{x})$ |
|---|---|---|---|---|---|
| 0 | 0 | 0 | -0.6703 | -0.8187 | 0.2256 |
| 0 | 0 | 1 | -0.6703 | -0.9048 | 0.2070 |
| 0 | 1 | 0 | -0.9048 | -0.9048 | 0.1637 |
| 0 | 1 | 1 | -0.9048 | -0.0067 | 0.4019 |
| 1 | 0 | 0 | -0.9048 | -0.8187 | 0.1784 |
| 1 | 0 | 1 | -0.9048 | -0.9048 | 0.1637 |
| 1 | 1 | 0 | -0.0067 | -0.9048 | 0.4019 |
| 1 | 1 | 1 | -0.0067 | -0.0067 | 0.9866 |
| $\sum \tilde{p}(\boldsymbol{x})$ | | | | | 2.7288 |

($\boldsymbol{x}$)

- **All three people are in good health**

$$p(1,1,1) = \frac{1}{2.7288} \times 0.9866 = 0.3616$$

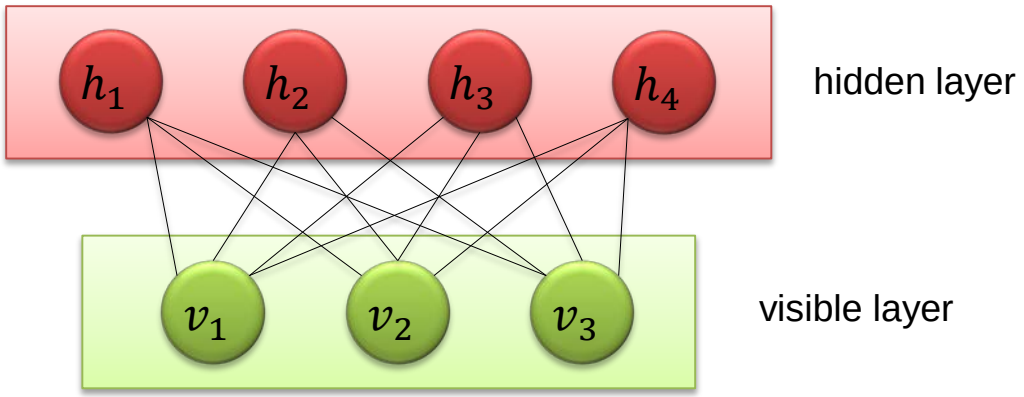
- **Roommate and you are coughing but coworker is fine**

$$p(0,0,1) = \frac{1}{2.7288} \times 0.2070 = 0.0759$$

# Restricted Boltzmann Machine (RBM)

- **RBM (Smolensky 1986) is quintessential example of how graphical models are used for deep learning**

- **RBM is an undirected, generative energy-based model with an input layer and single hidden layer**
  - connections only exist between the visible units of the input layer and the hidden units of the hidden layer
  - there are no visible-visible or hidden-hidden connections



# Energy of a State $(\boldsymbol{v}, \boldsymbol{h})$

$\boldsymbol{b}, \boldsymbol{c}, \boldsymbol{W}$: learnable parameters
$\boldsymbol{v}$: visible variables
$\boldsymbol{h}$: latent variables

$$E(\boldsymbol{v}, \boldsymbol{h}) = -\boldsymbol{b}^{\top}\boldsymbol{v} - \boldsymbol{c}^{\top}\boldsymbol{h} - \boldsymbol{v}^{\top}\boldsymbol{W}\boldsymbol{h}$$

- **Weights → Energy → Probability**
  - each possible state $(\boldsymbol{v}, \boldsymbol{h})$ has an energy
  - the energy is determined by the parameters

- **The energy of a state determines its probability**

$$p(\boldsymbol{v}, \boldsymbol{h}) \propto e^{-E(\boldsymbol{v}, \boldsymbol{h})}$$

$$p(\boldsymbol{h}|\boldsymbol{v}) = \prod_{\boldsymbol{i}} p(h_i|\boldsymbol{v})$$

$$p(\boldsymbol{v}|\boldsymbol{h}) = \prod_{\boldsymbol{i}} p(v_i|\boldsymbol{h})$$

# Training RBMs

- **Maximizing the product of probabilities assigned to some given training set $\boldsymbol{V}$:**

$$arg\max_{\boldsymbol{W}} \prod_{\boldsymbol{v}\in\boldsymbol{V}} p(\boldsymbol{v};\boldsymbol{W})$$

  - equivalent to maximizing the *log-likelihood*

$$arg\max_{\boldsymbol{W}} \sum_{\boldsymbol{v}\in\boldsymbol{V}} \log p(\boldsymbol{v};\boldsymbol{W})$$

- **Gradient descent method for training RBMs:**

$$\frac{d\log(p(\boldsymbol{v}))}{d\boldsymbol{W}} = E_{P_{data}}[\boldsymbol{v}\boldsymbol{h}^{\top}] - E_{P_{model}}[\boldsymbol{v}\boldsymbol{h}^{\top}]$$

  - $E_p$: expectation over given mini-batch

# Contrastive Divergence (CD)

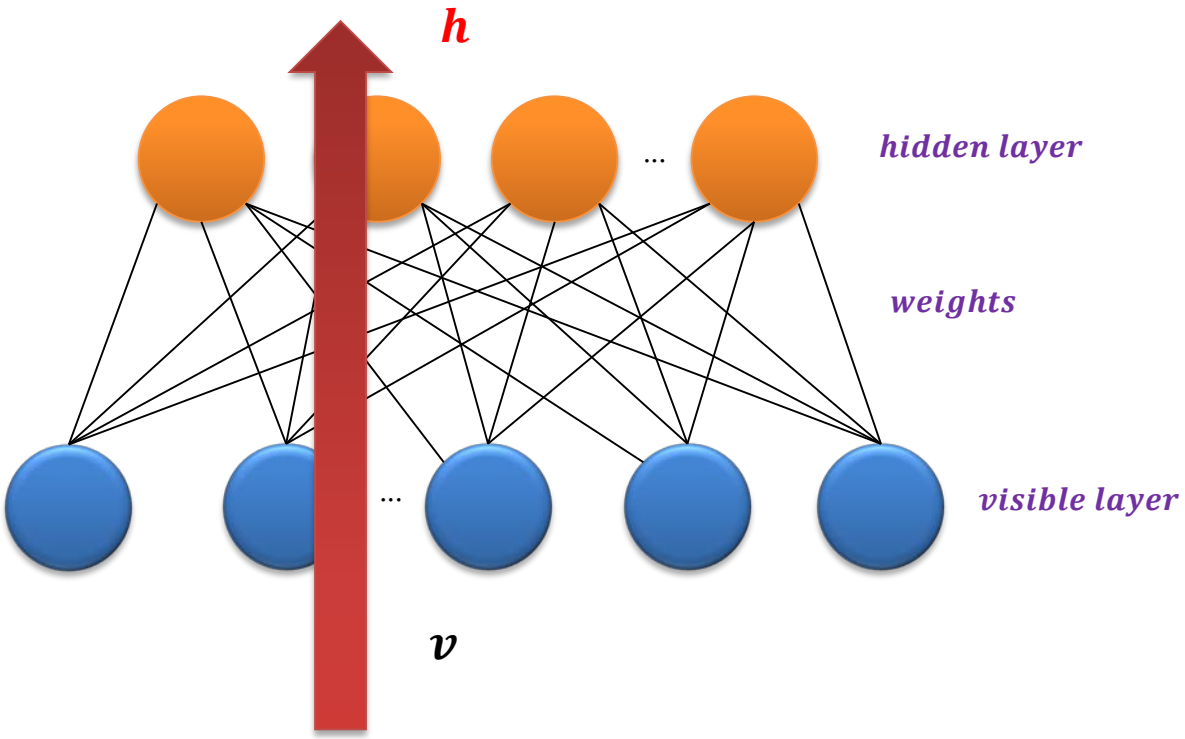
- **Positive step:**
  - an input $\boldsymbol{v}$ is clamped to the input layer
  - $\boldsymbol{v}$ is propagated to the hidden layer (like NN)
  - the result of the hidden layer activations is $\boldsymbol{h}$
- **Negative step:**
  - propagate $\boldsymbol{h}$ back to the visible layer with result $\boldsymbol{v}'$
  - propagate $\boldsymbol{v}'$ back to the hidden layer with result $\boldsymbol{h}'$
- **Weight update:**

$$\boldsymbol{W}^{(t+1)} = \boldsymbol{W}^{(t)} + a(\boldsymbol{v}\boldsymbol{h}^{\mathsf{T}} - \boldsymbol{v}'\boldsymbol{h}'^{\mathsf{T}})$$

  - $a$: learning rate

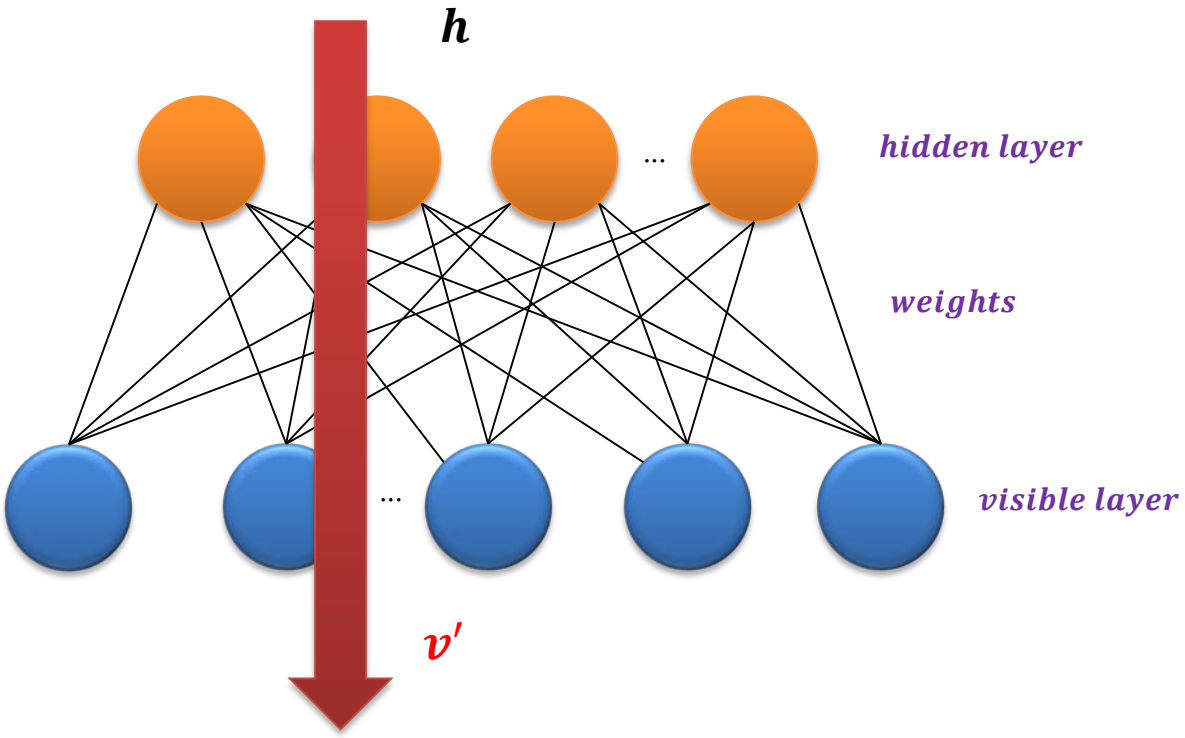
# CD: Positive Step

- **For given input data $v$,**
  - ➢ obtain the activations of the hidden layer $h$



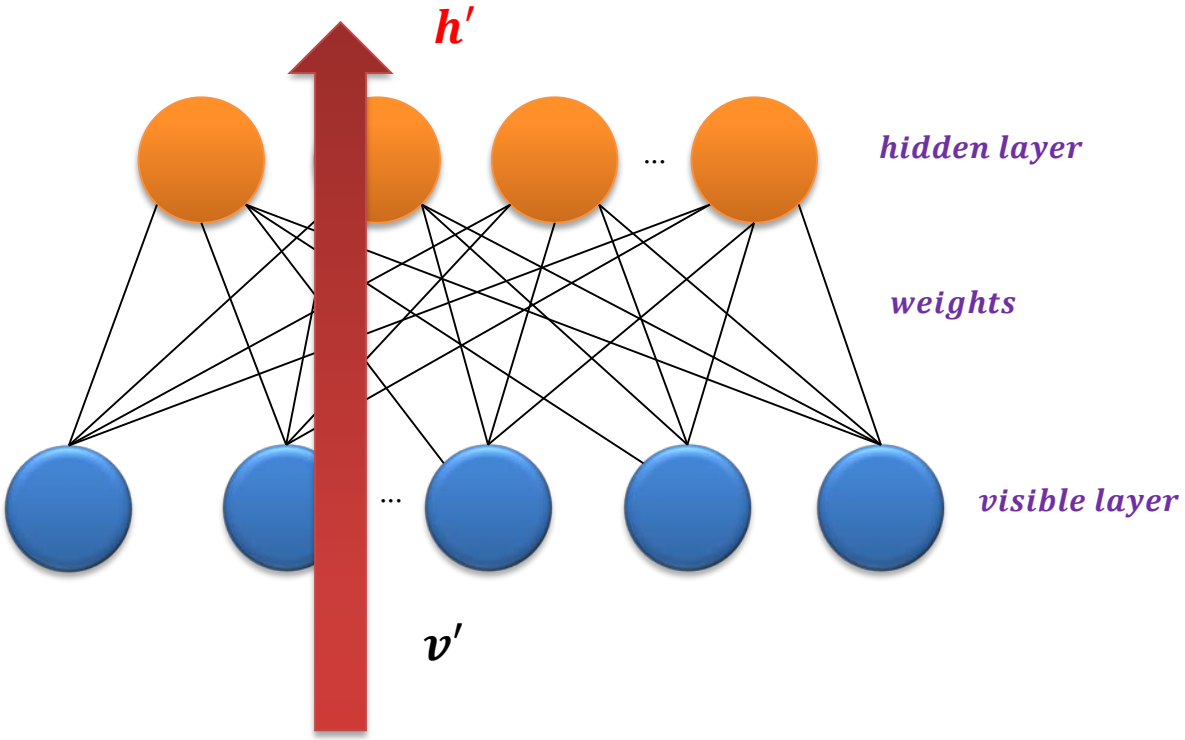
# CD: Negative Step (Backward)

- **For given hidden data $h$,**
  - recreate the activation of visible layer $v'$



# CD: Negative Step (Forward)

- **Again, for given visible data $v'$,**
  - ➢ obtain the activations of the hidden layer $h'$



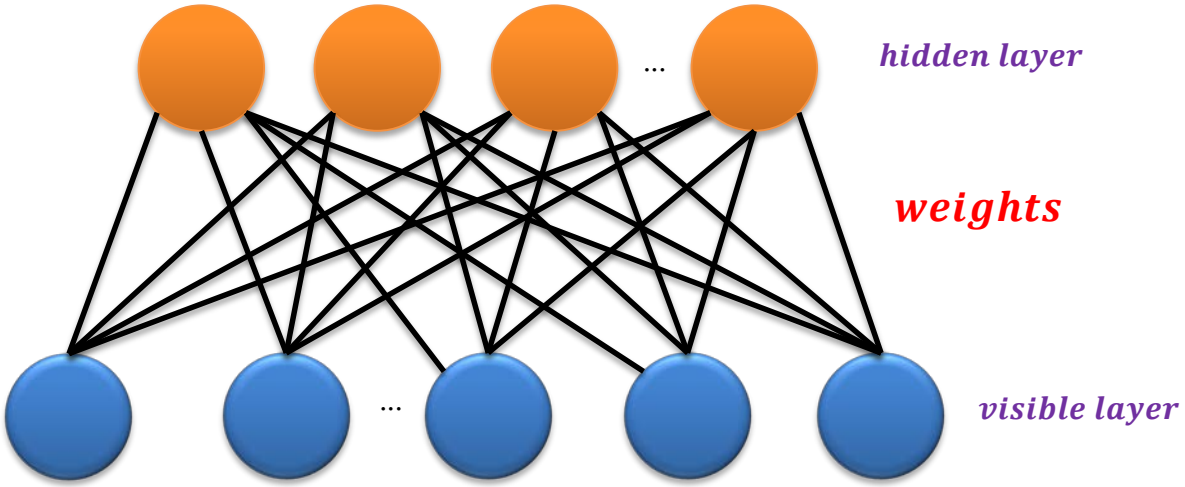
# CD: Weight Update

- **Using the activations obtained in single CD iteration,**
  - ➢ update the weights:

positive step

negative step

$$\boldsymbol{W}(t+1) = \boldsymbol{W}(t) + a(\boldsymbol{v}\boldsymbol{h}^{\top} - \boldsymbol{v}'\boldsymbol{h}'^{\top})$$



# Goal of Contrastive Divergence

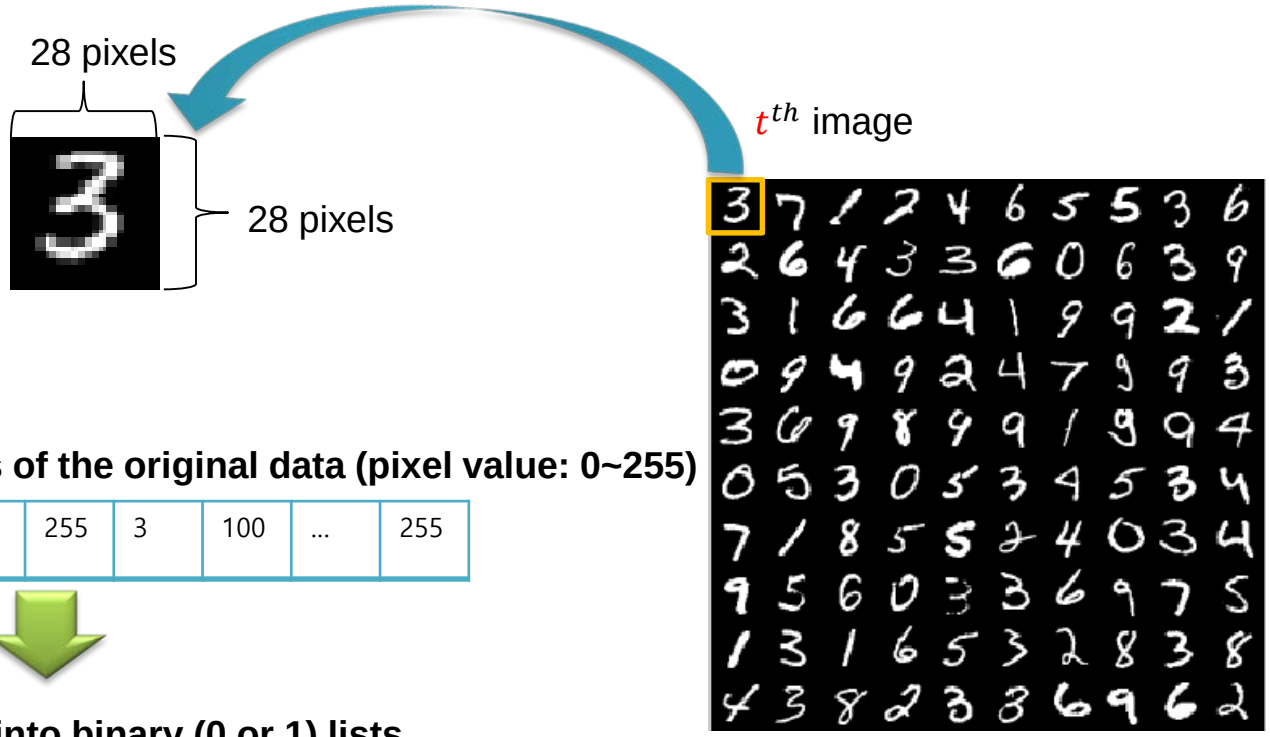
- *Positive phase* **reflects the network's internal representation of the real world data**

- *Negative phase* **represents an attempt to recreate the data based on this internal representation**

- **Main goal is for the generated data to be as close as possible to the real world**
  - this is reflected in the weight update formula

# Training Binary-RBM with MNIST

- **Preparing an visible data, $posV$**




**An array of pixel values of the original data (pixel value: 0~255)**

| 29 | 250 | 10 | 36 | 255 | 255 | 3 | 100 | ... | 255 |
|---|---|---|---|---|---|---|---|---|---|

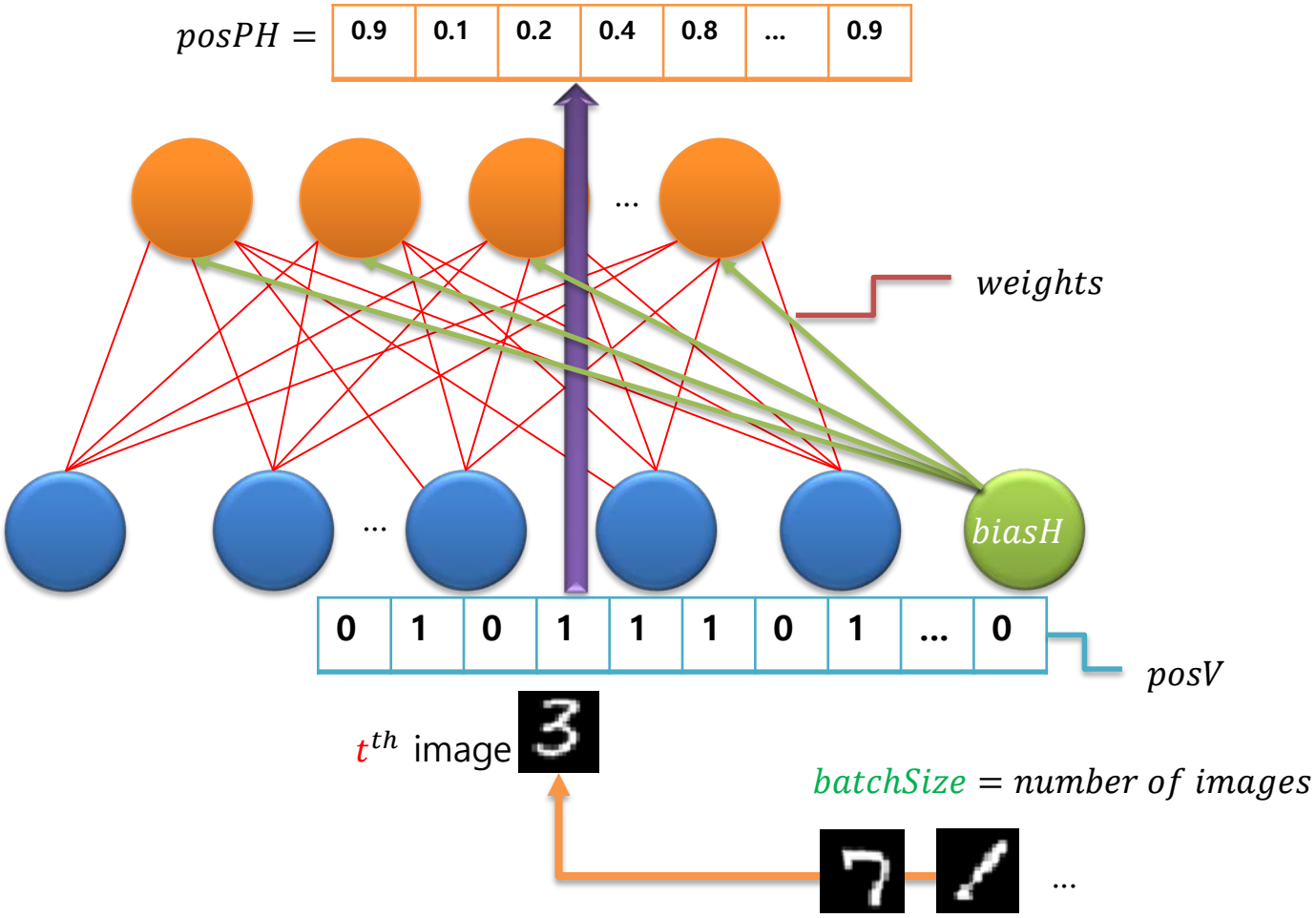
**Image data converted into binary (0 or 1) lists**

$posV =$

| 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | ... | 0 |
|---|---|---|---|---|---|---|---|---|---|

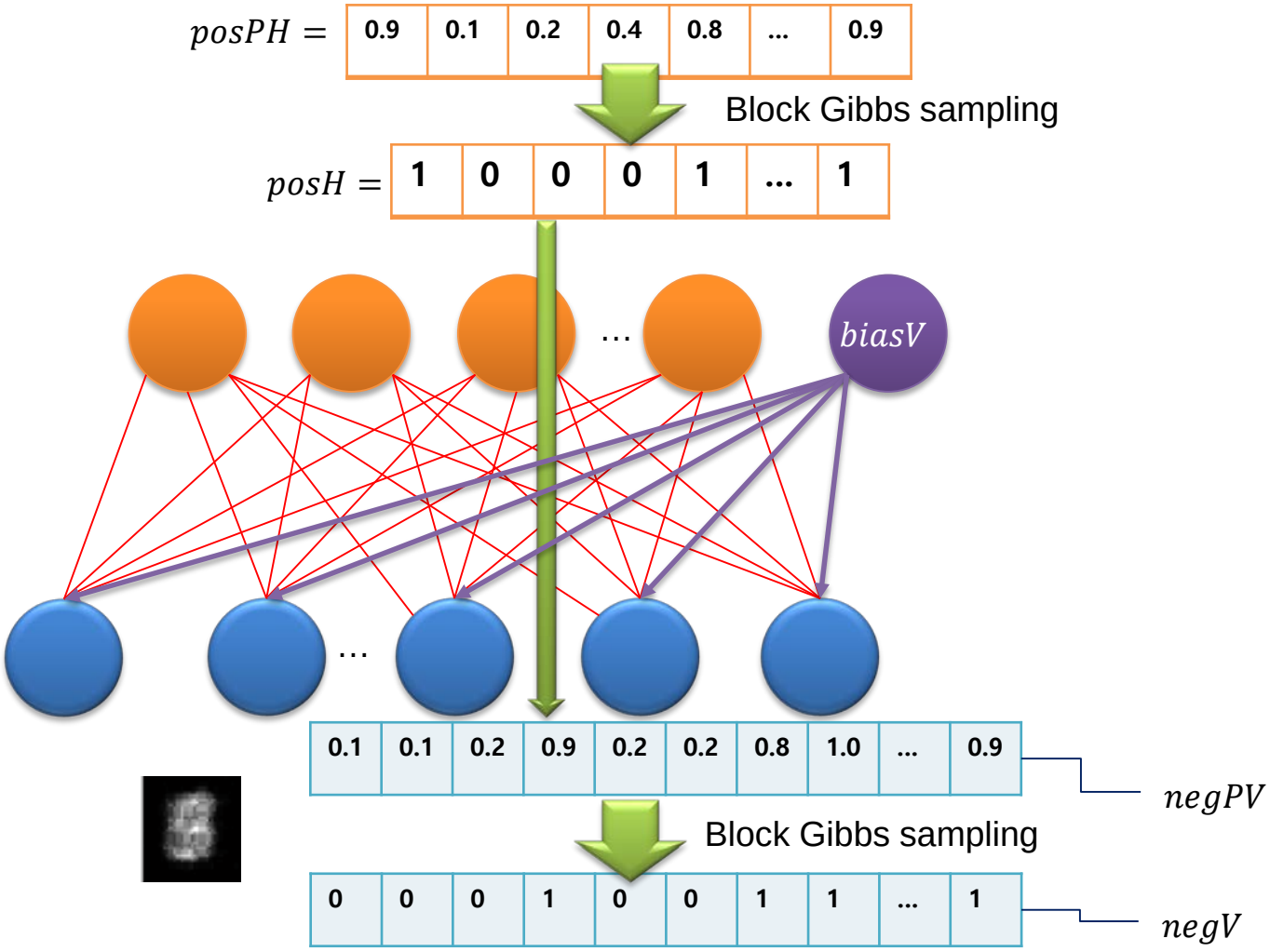
* For ease of understanding, the detailed values in the lists are arbitrary numbers

# Positive Phase



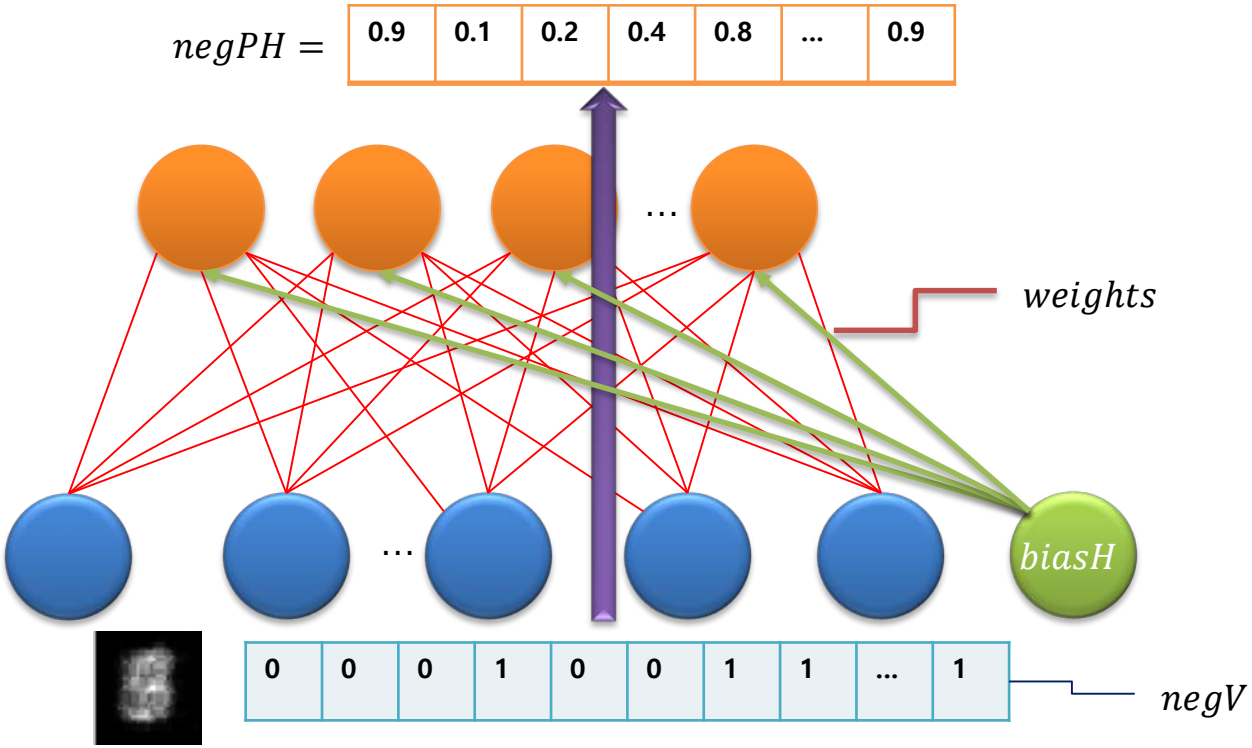

$posPH < 1 \times nHid >$
$biasH < 1 \times nHid >$
$posV < 1 \times nVis >$
$weights < nHid \times nVis >$

# Negative Phase



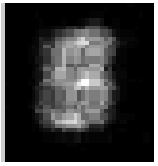

$negPH < 1 \times nHid >$
$negH < 1 \times nHid >$
$biasV < 1 \times nVis >$
$posPH < 1 \times nHid >$
$posH < 1 \times nHid >$
$weights < nHid \times nVis >$

# Negative Phase




$negPH < 1 \times nHid >$
$biasH < 1 \times nHid >$
$negV < 1 \times nVis >$
$weights < nHid \times nVis >$

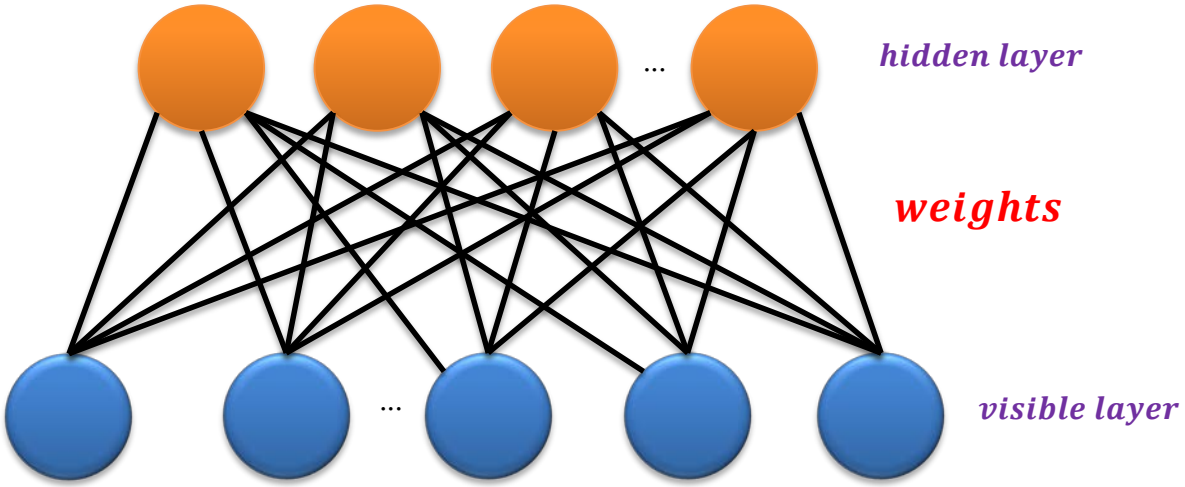
# Weight Update



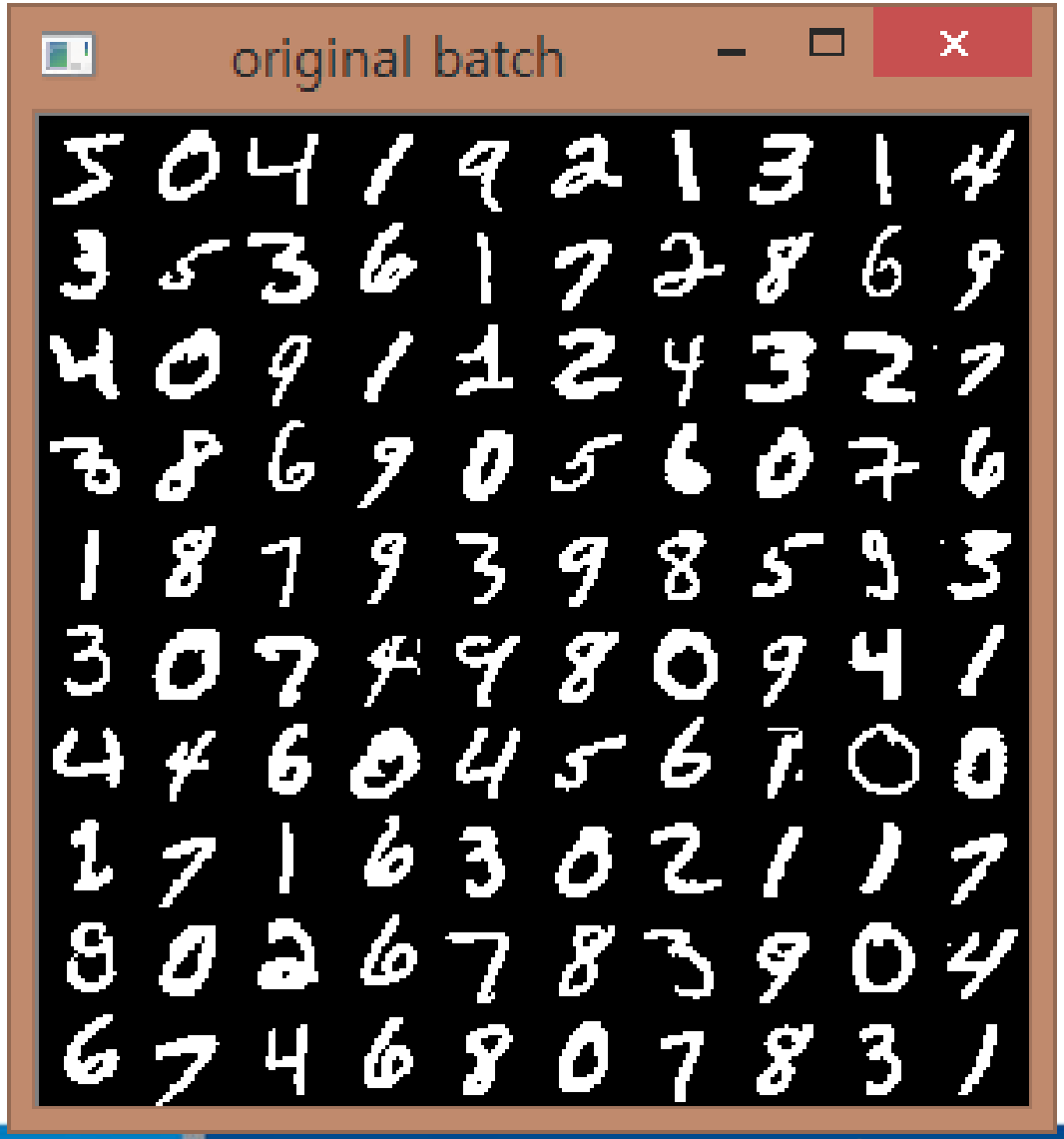
positive step

negative step

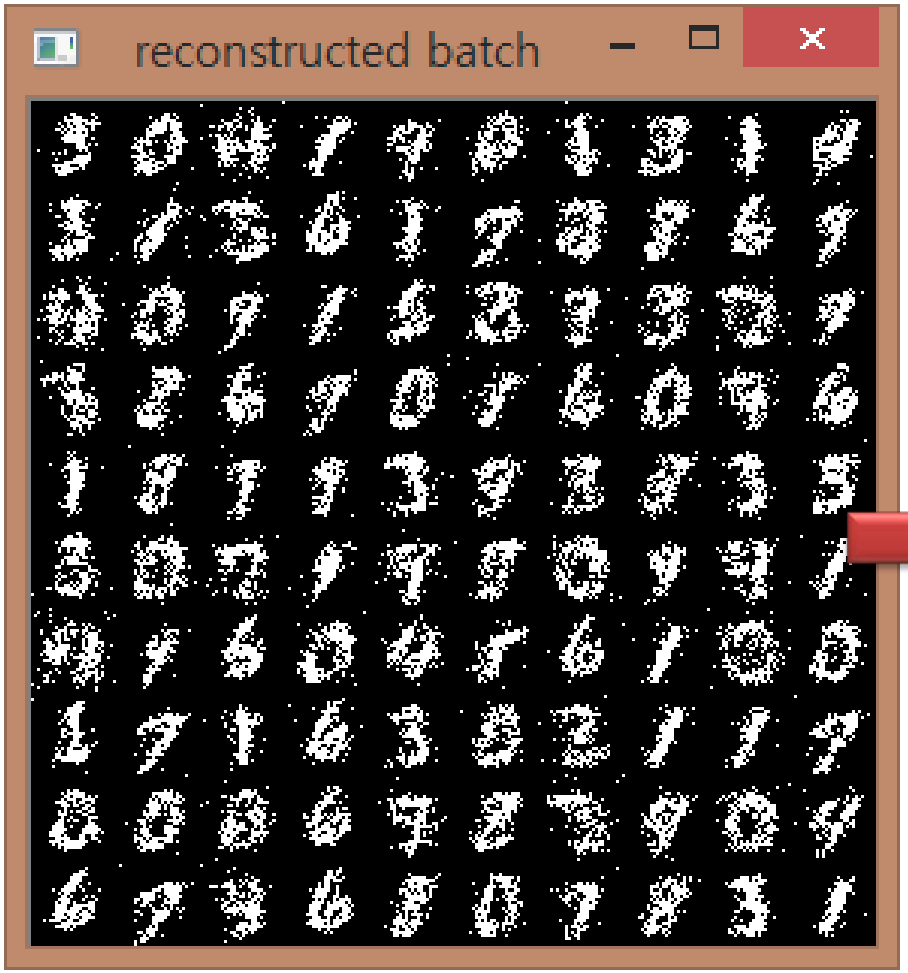
$$\boldsymbol{W}(t+1) = \boldsymbol{W}(t) + a(\boldsymbol{v}\boldsymbol{h}^{\top} - \boldsymbol{v}'\boldsymbol{h}'^{\top})$$



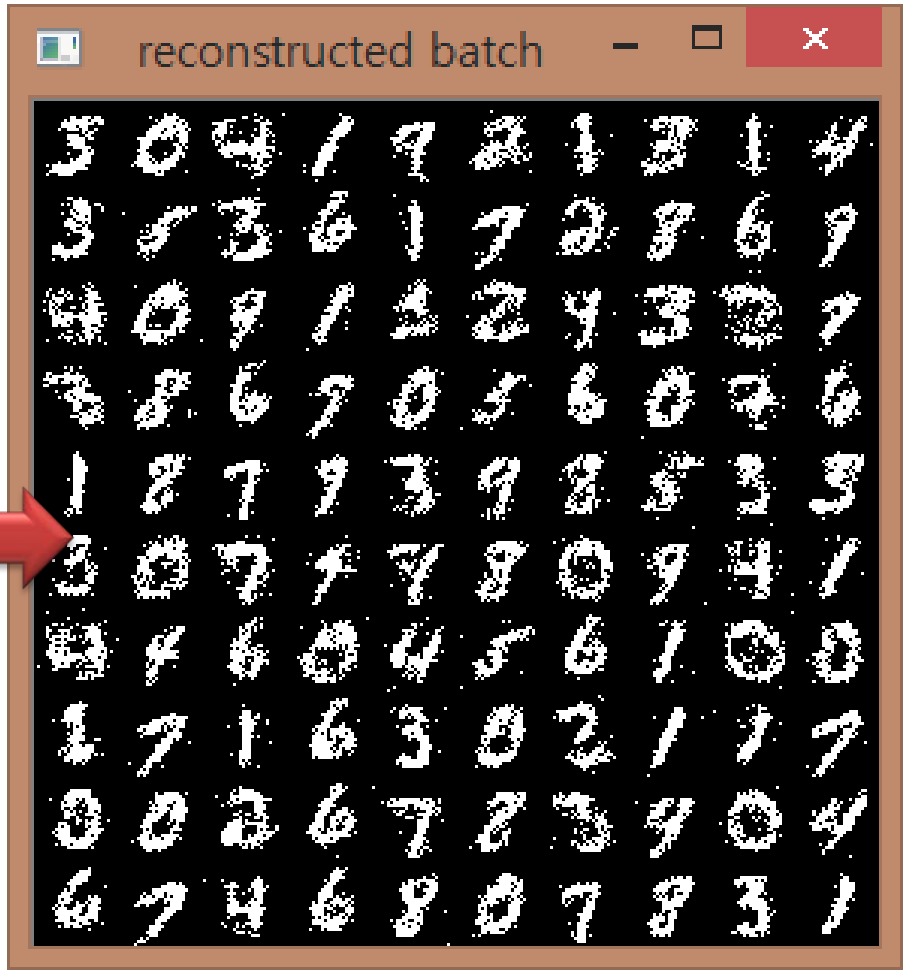
# Original Training Images



# Training Result



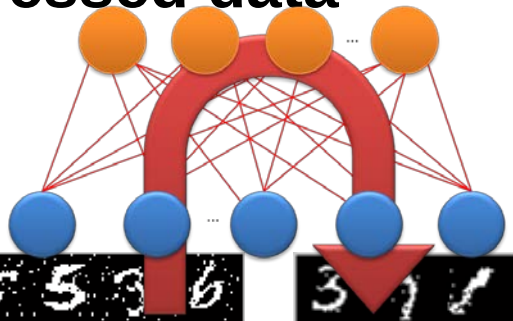

epoch 1000

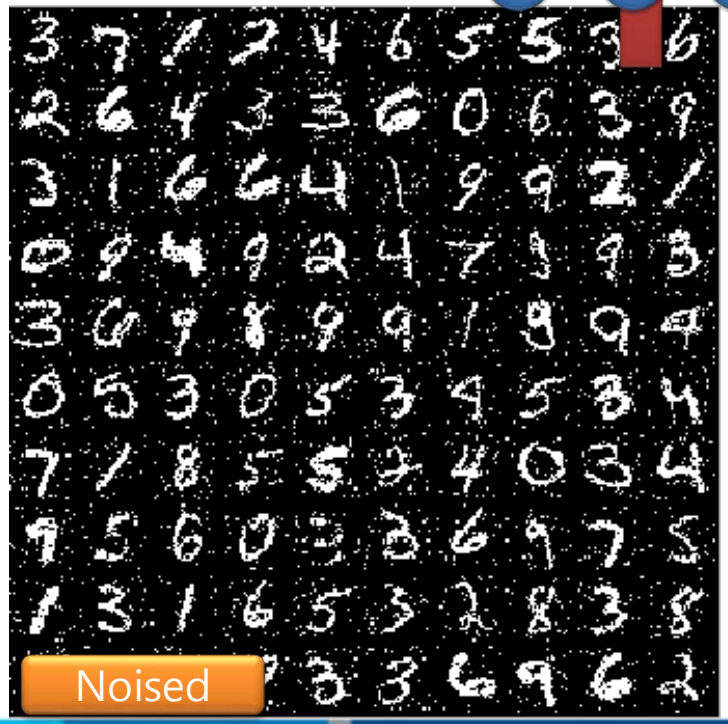


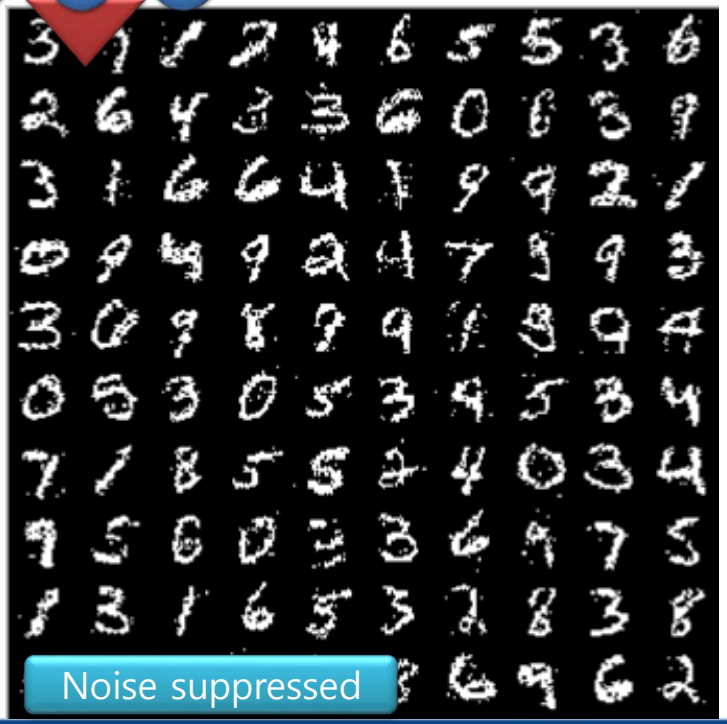

epoch 3000

# Testing Result

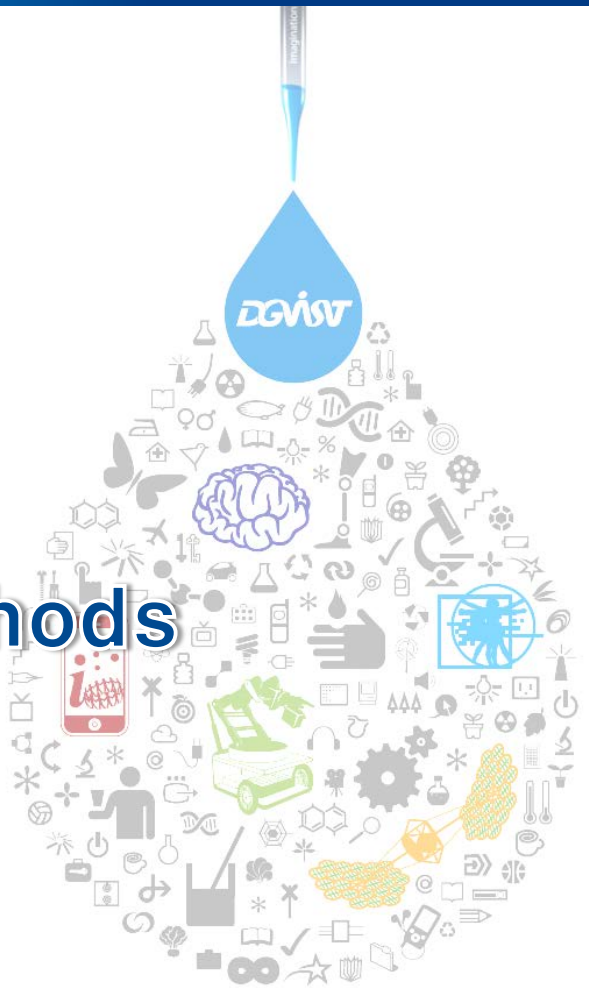
- Input: noised batch data
- Output: noise suppressed data





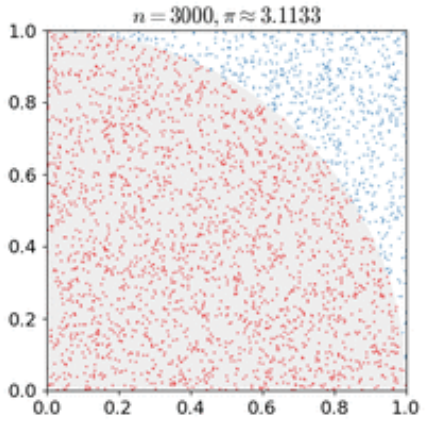
# Chapter 17. Monte Carlo Methods



# Two Sorts of Randomized Algorithms

- **Las Vegas algorithms**
  - always return precisely the correct answer
  - consume a random amount of resources (memory or time)
  - e.g., quicksort

- **Monte Carlo algorithms**
  - return answers with a random amount of error
  - the amount of error can typically reduced by expending more resources
  - e.g., approximating PI



# Basics of Monte Carlo Methods

- **To view the sum/integral as if it were an expectation under some distribution**

$$s = \sum_{\boldsymbol{x}} p(\boldsymbol{x}) f(\boldsymbol{x}) = E_p[f(\boldsymbol{x})]$$

$$s = \int p(\boldsymbol{x}) f(\boldsymbol{x}) d\boldsymbol{x} = E_p[f(\boldsymbol{x})]$$

- **To approximate the expectation by a corresponding average**
  - drawing $n$ samples $\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(n)}$ from $p$
  - forming the empirical average:

$$\hat{s}_n = \frac{1}{n} \sum_{i=1}^{n} f(\boldsymbol{x}^{(i)})$$

  - by the law of large number, if the sample $\boldsymbol{x}^{(i)}$ are i.i.d.,

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

# Markov Chain Monte Carlo (MCMC)

- **Markov chain is a stochastic model describing a sequence of possible events**
  - probability of each event depends only on the state attained in the previous event
  - Markov assumption: memorylessness

$$P(X_t|X_{t-1}, X_{t-2}, \ \dots \ , X_1) = P(X_t|X_{t-1})$$

- **MCMC is the method for sampling from target probability distributions using Markov chains**
  - the family of algorithm that use Markov chains to perform Monte Carlo estimates

- **Advantages:**
  - converge to a stationary (or equilibrium) distribution
  - not depend on the initial state

# Gibbs Sampling (1984)

- **Gibbs sampling is a MCMC algorithm for obtaining a sequence of observations**
  - approximated from a **specified multivariate probability distribution** when direct sampling is difficult
- **Gibbs sampling generates a Markov chain of samples**
  - each of which is correlated with **nearby** samples
  - samples from the beginning of the chain may not accurately represent the desired distribution

# Gibbs Sampling Algorithm

- **The point of Gibbs sampling is that given a multivariate distribution**

- **It is simpler to sample from a conditional distribution than to marginalize by integrating over a joint distribution**

  - $Let\ p(\boldsymbol{x})$: $pdf$ of $\boldsymbol{x}$, $\boldsymbol{x} = (x_1, x_2, \dots, x_m)$, we want to obtain $k$ samples

$init$: $\boldsymbol{x}^{(0)}$ $with\ some\ value$

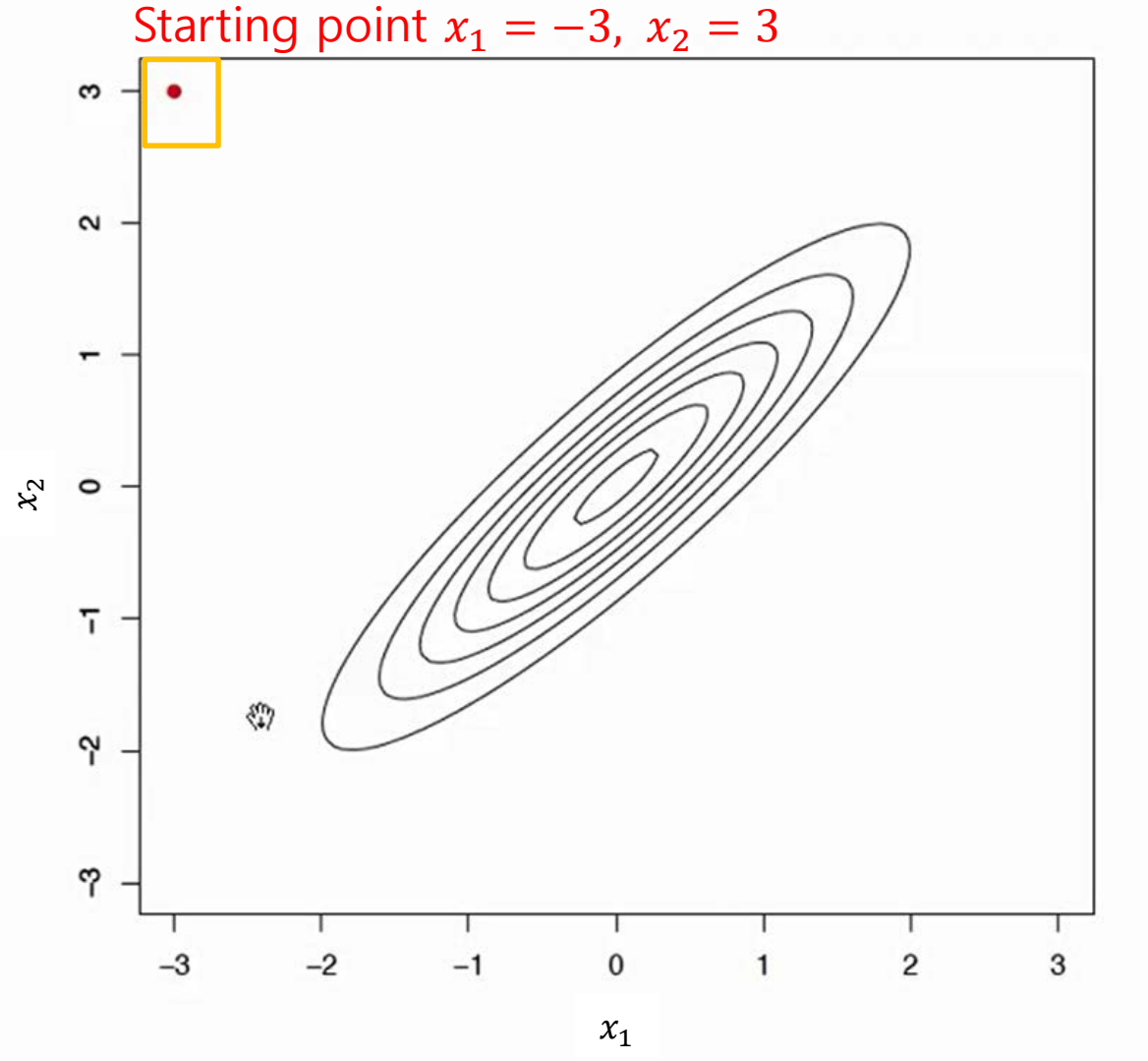
$for\ i = 1{:}k$

$for\ j = 1{:}m$

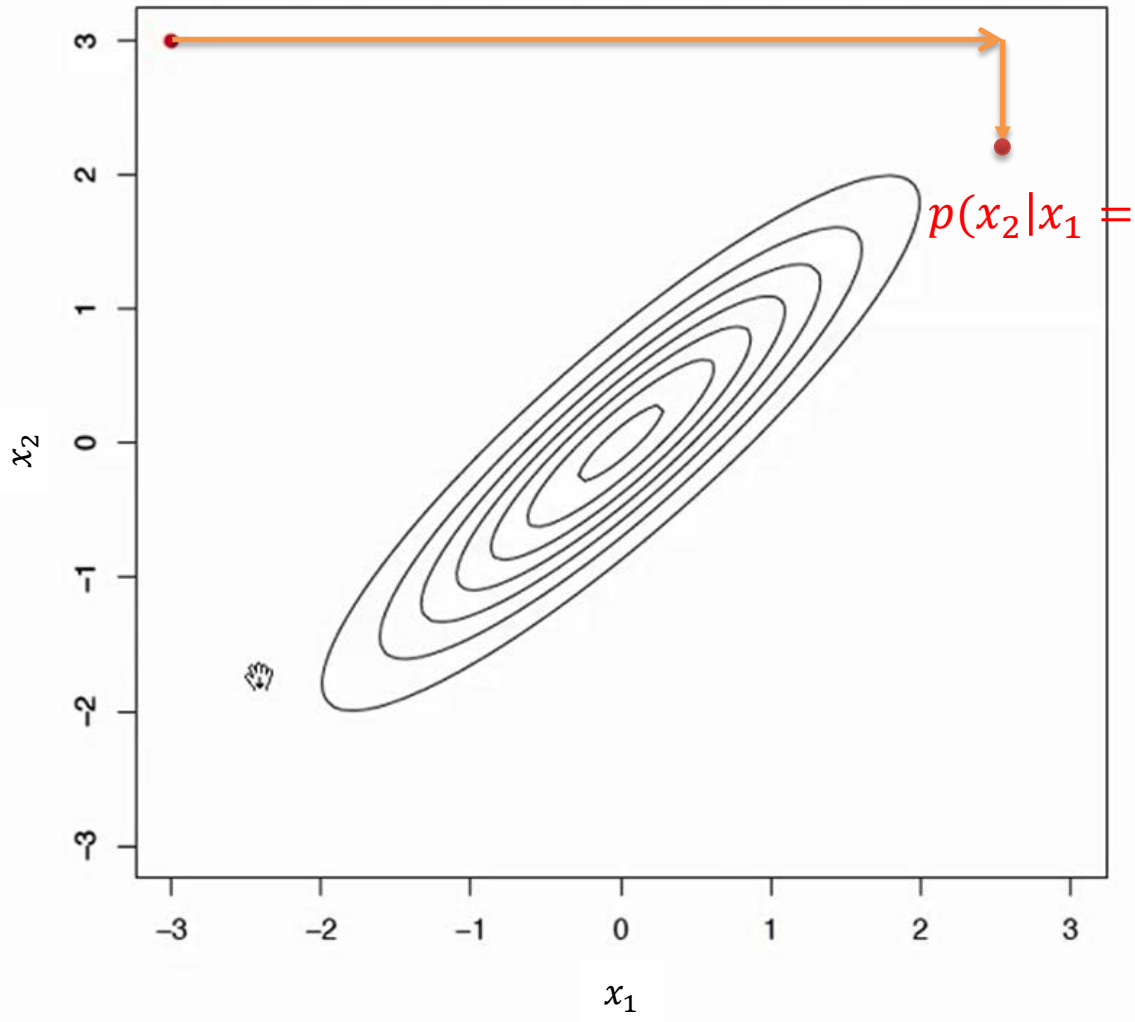
$draw\ x_j^{(i)}$ from $p\left(x_j | x_1^{(i)}, \dots x_{j-1}^{(i)}, x_{j+1}^{(i-1)}, x_m^{(i-1)}\right)$

full conditional distributions

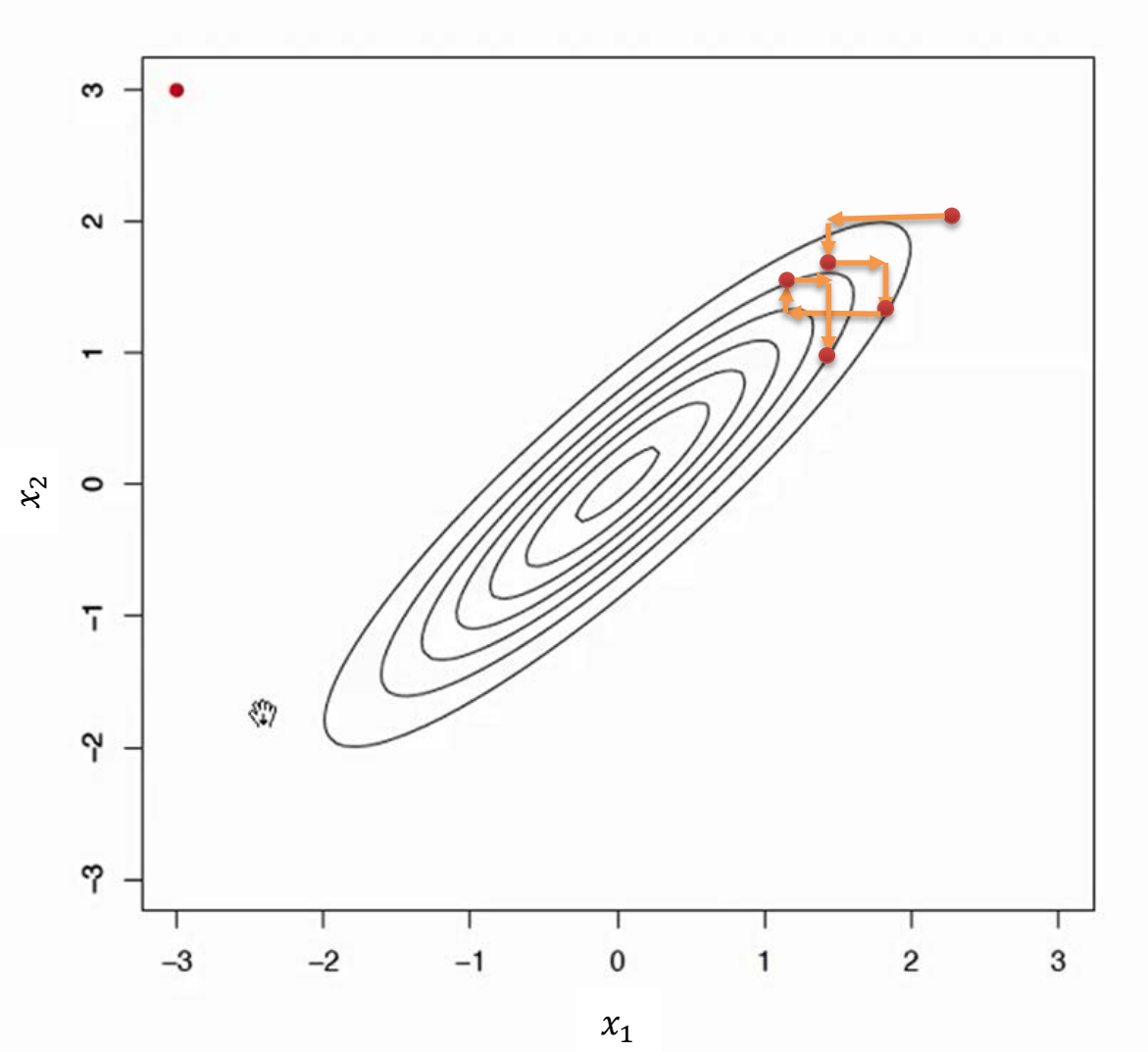
# Example: Drawing $x = (x_1, x_2)$



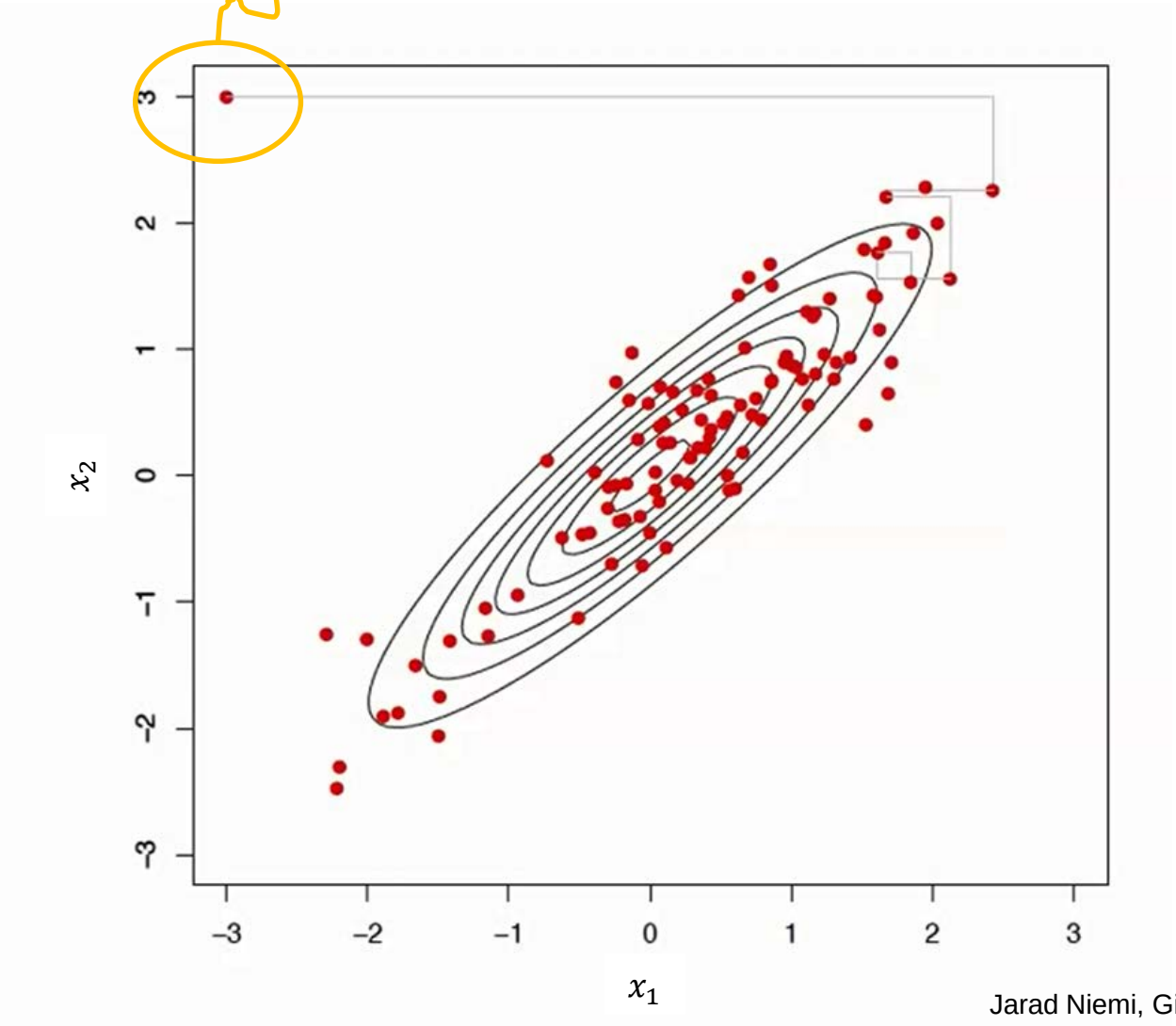
$p(x_1|x_2 = 3) \rightarrow x_1 = 2.3$



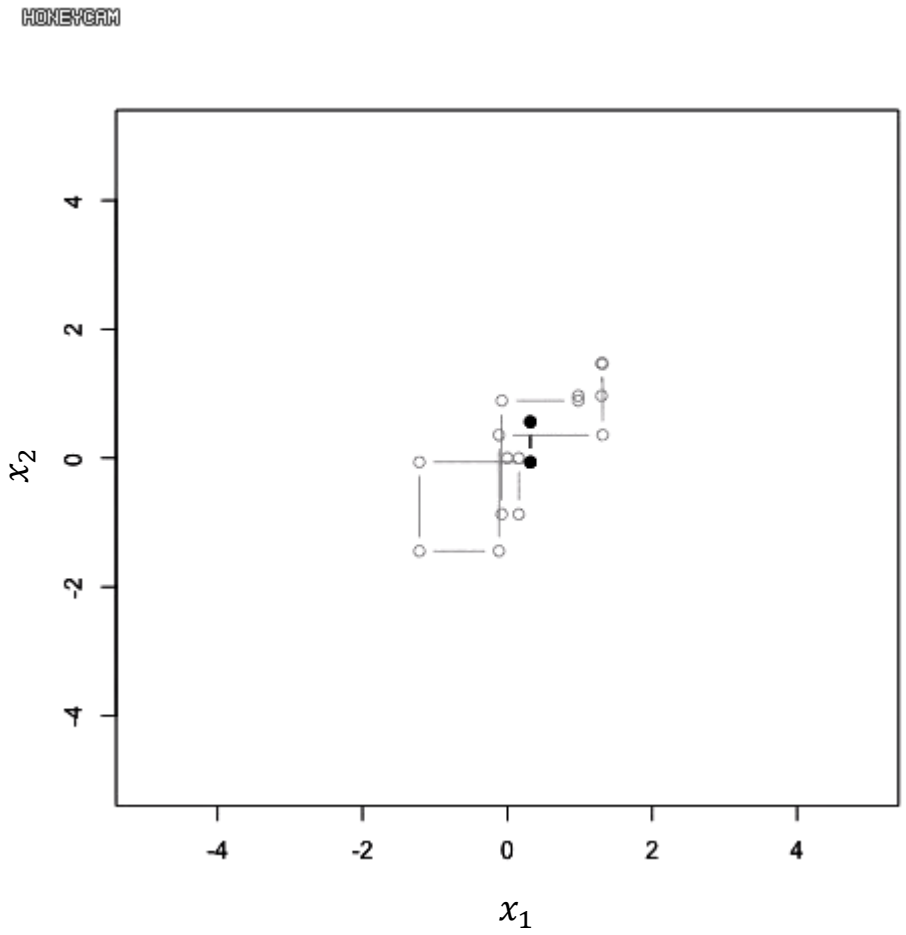

$p(x_2|x_1 = 2.3) \rightarrow x_2 = 2$


3
2
1
0
−1
−2
−3
$x_2$
−3
−2
−1
0
1
2
3
$x_1$

(usually) remove the first point




Jarad Niemi, Gibbs sampling, 2013
https://youtu.be/a_08GKWHFWo?t=259




https://ratsgo.github.io/statistics/2017/05/31/gibbs/

# Q&A

Thank you