Deep Learning Seminar
Chapter 20. Deep Generative Models

Keonwoo Noh
Department of Information and Communication Engineering
DGIST

Chapter 20. Deep Generative Models

Chapter 20. Deep Generative Models

- 20.9 Back-Propagation through Random Operations
- 20.10 Directed Generative Nets
- 20.11 Drawing Samples from Autoencoders
- 20.12 Generative Stochastic Networks
- 20.13 Other Generation Schemes
- 20.14 Evaluating Generative Models
- 20.15 Conclusion

Chapter 20. Deep Generative Models

- Generative models
- Boltzmann Machines
- Restricted Boltzmann Machines
- Deep Belief Networks

Generative models

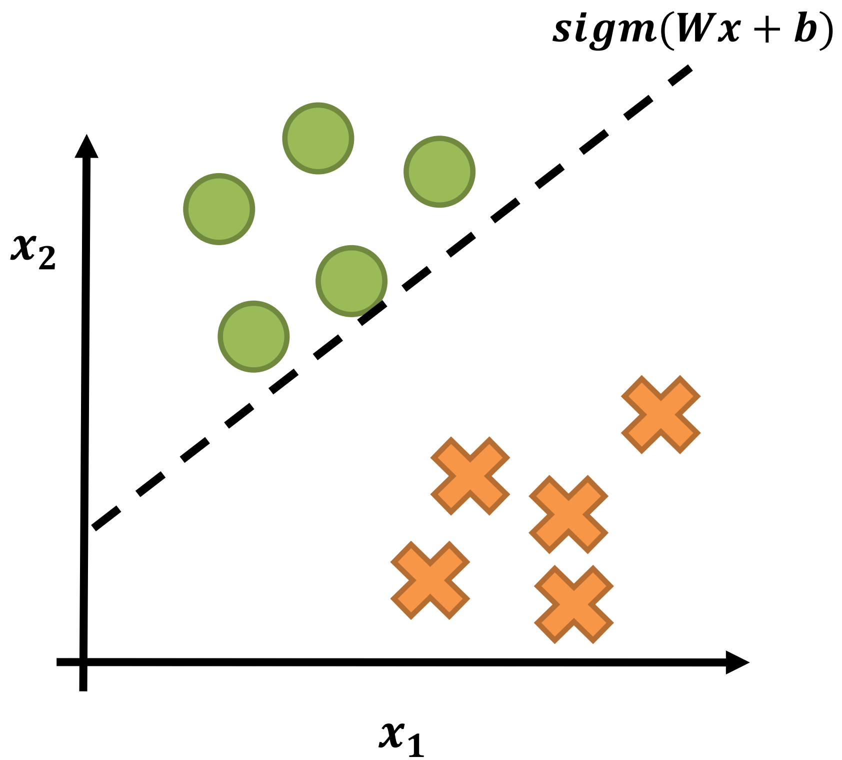
Generative V.S. Discriminative

- **Generative**
  - To model the joint probability and to make a decision using the result of **'generate' the samples** into the distribution
  - E.g. Autoencoders, Restricted Boltzmann Machines
- **Discriminative**
  - **To directly compute the posterior class probability** $p(C|X)$ in the inference stage
  - E.g. Logistic regression, SVM, Boosting, Neural networks

$$\boxed{p(C|X)} = \frac{\boxed{p(X|C)p(C)}}{p(X)}$$

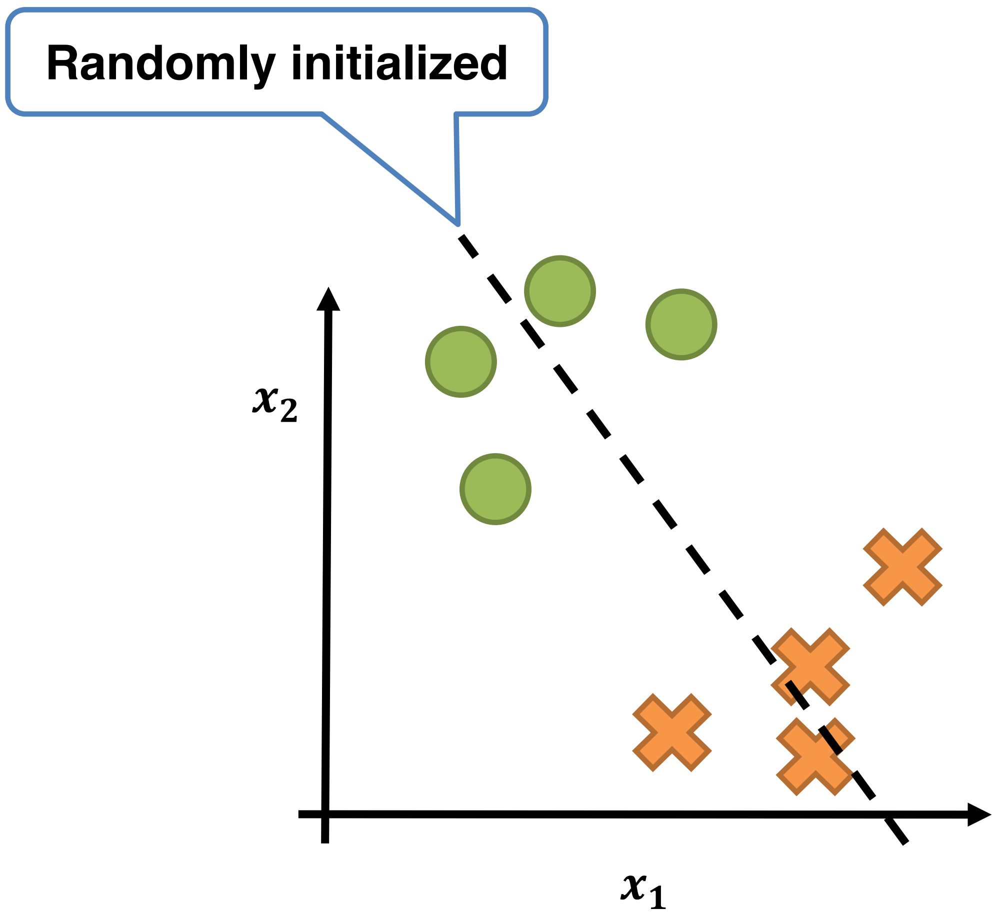
Discriminative Model – Logistic Regression

- Classification model
- Finding decision boundary that minimize error rate



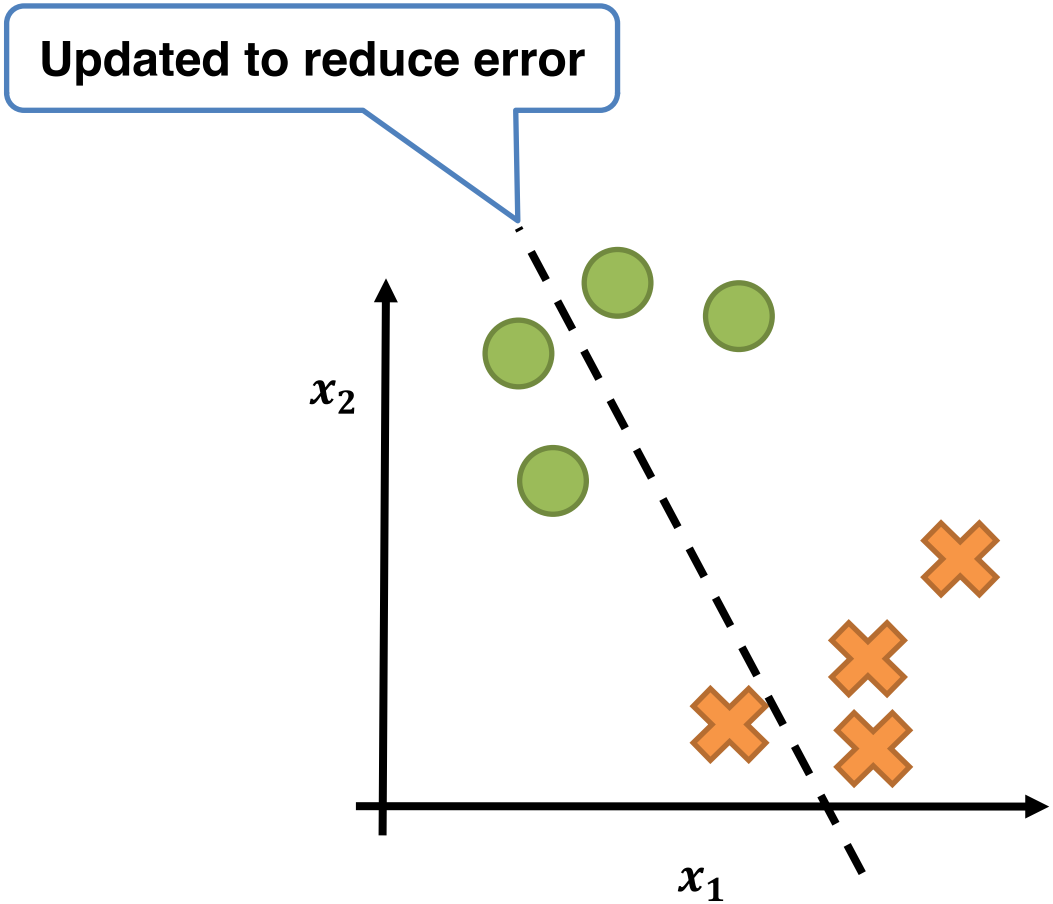
An Example of Logistic Regression

- **Decision boundary is randomly initialized**



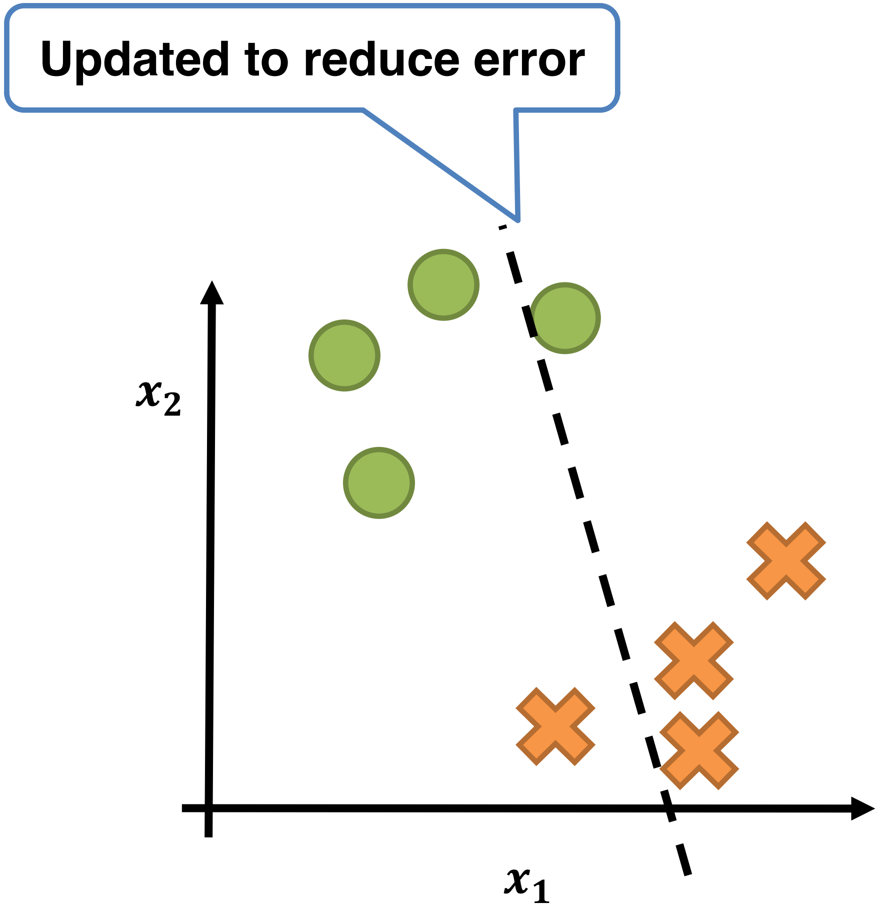
An Example of Logistic Regression

- **Update parameters of decision boundary**



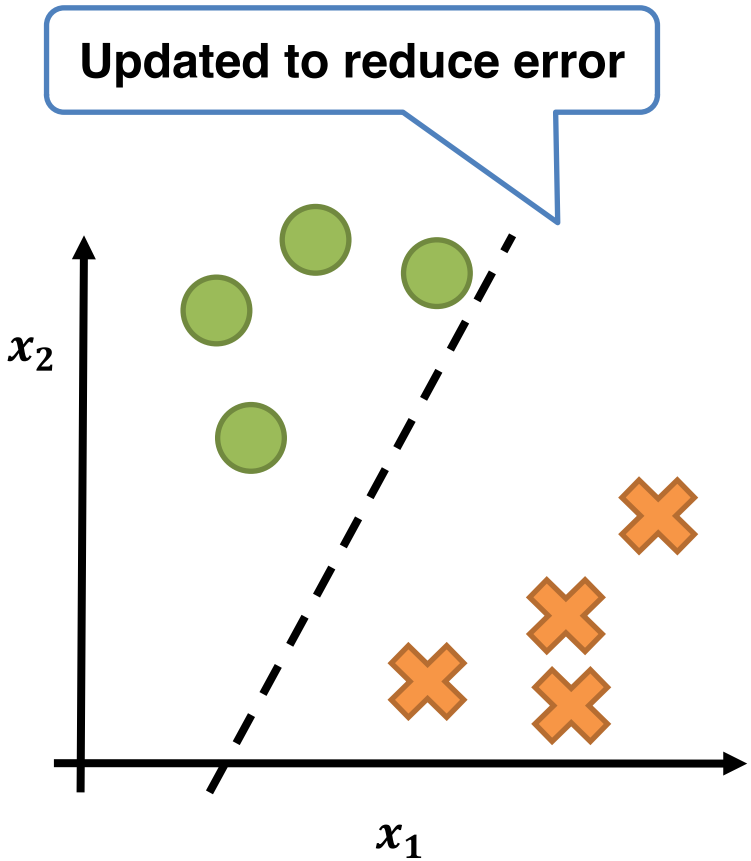
An Example of Logistic Regression

- **Update parameters of decision boundary**



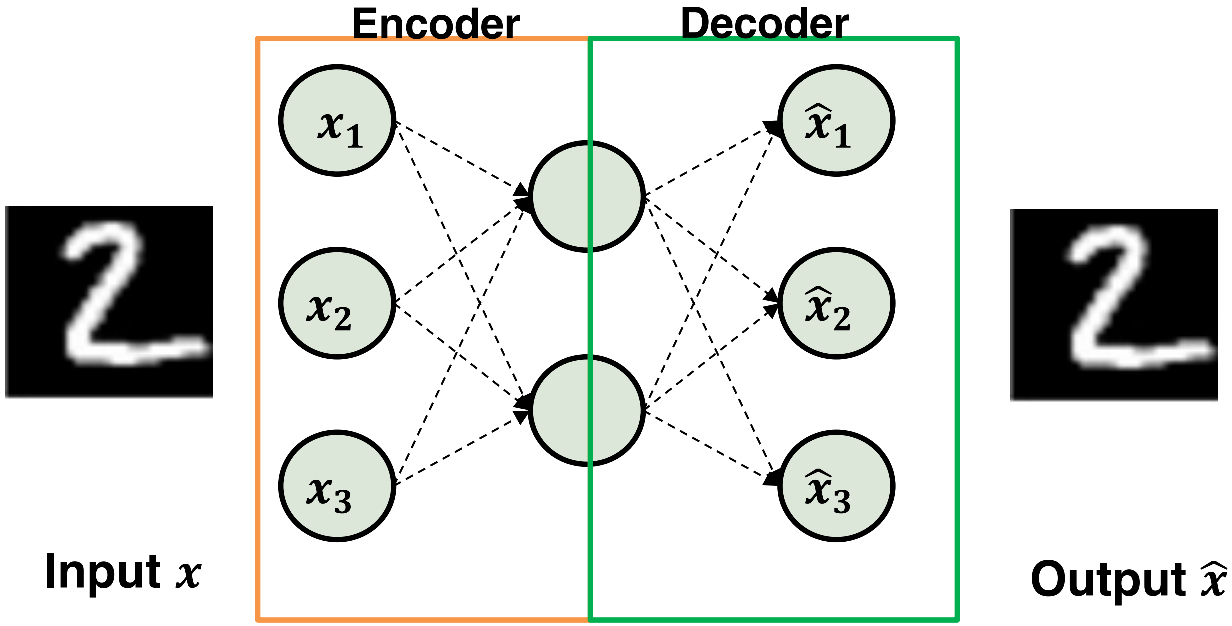
An Example of Logistic Regression

- **Update parameters of decision boundary**



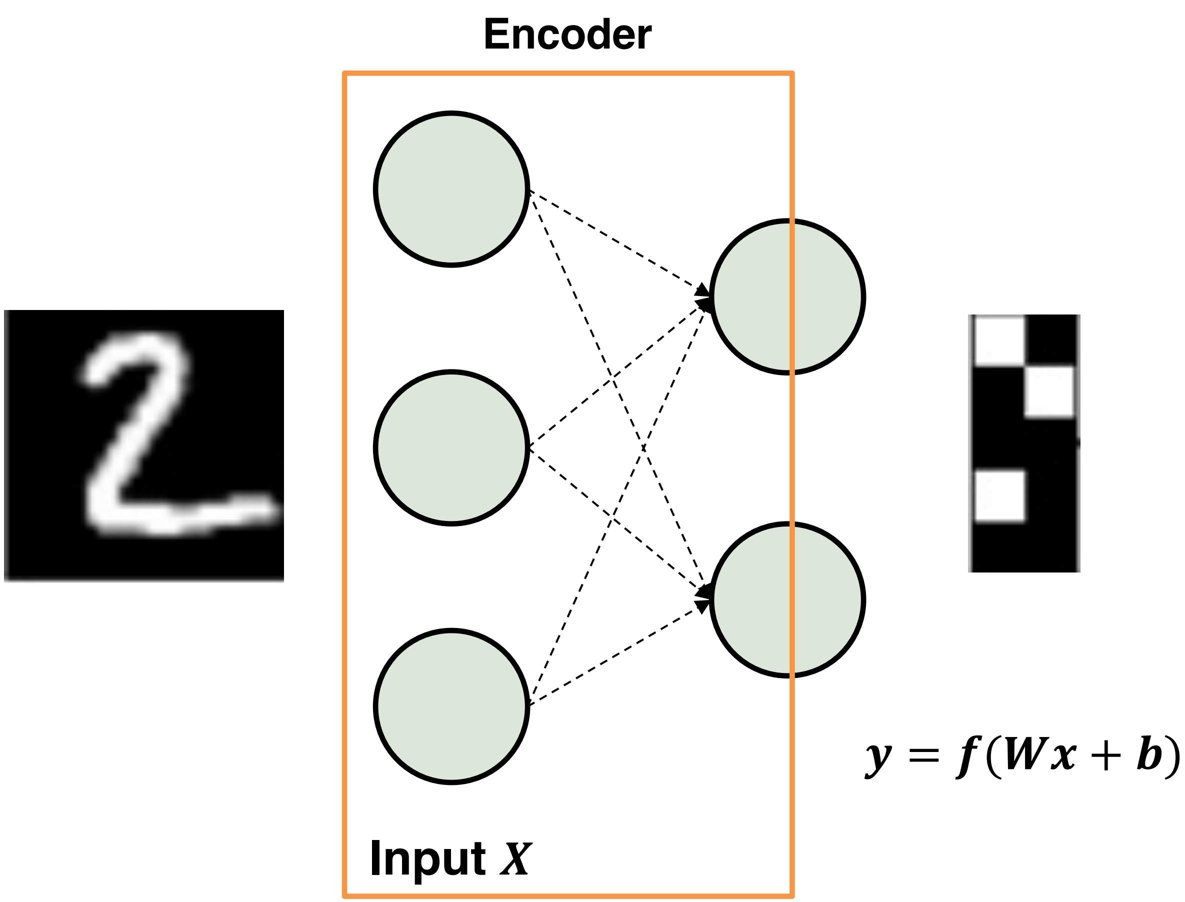
Generative model – AutoEncoders (AE)

- AE is neural network models that 'generate' the same data as the input data
- The main goal of AE is to train distribution of data
- AE with sigmoidal hidden units and linear reconstruction unit is equivalent to RBM



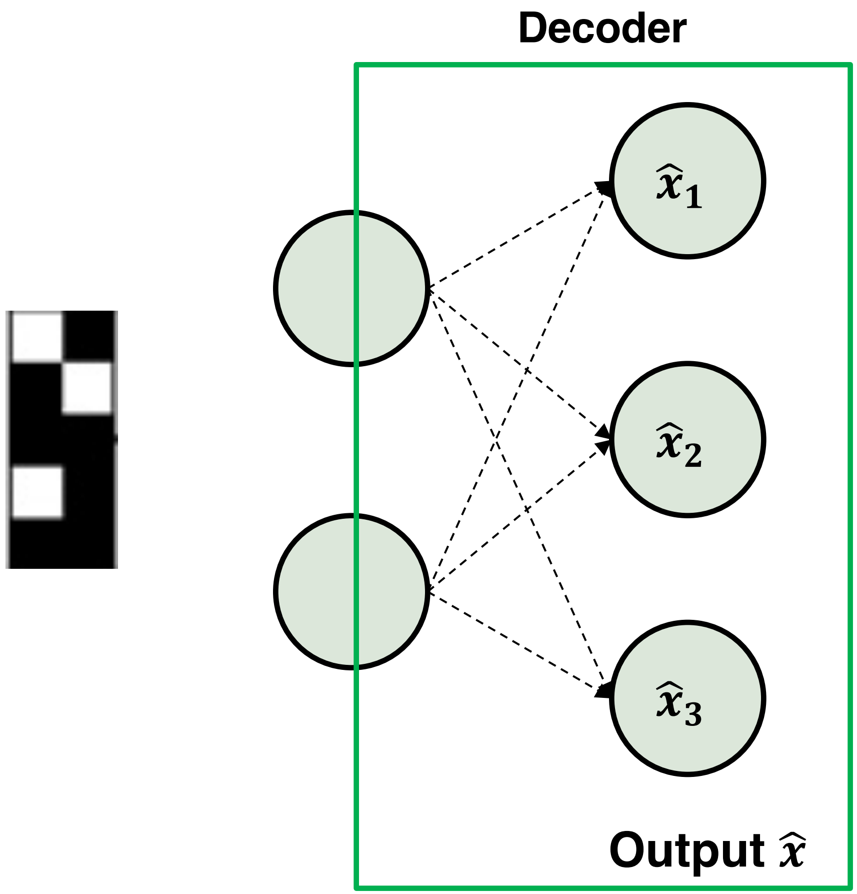
An Example of AE – Encoder

- **Encoder yields compression of input data**



An Example of AE – Decoder

- **Decoder reconstructs input data**

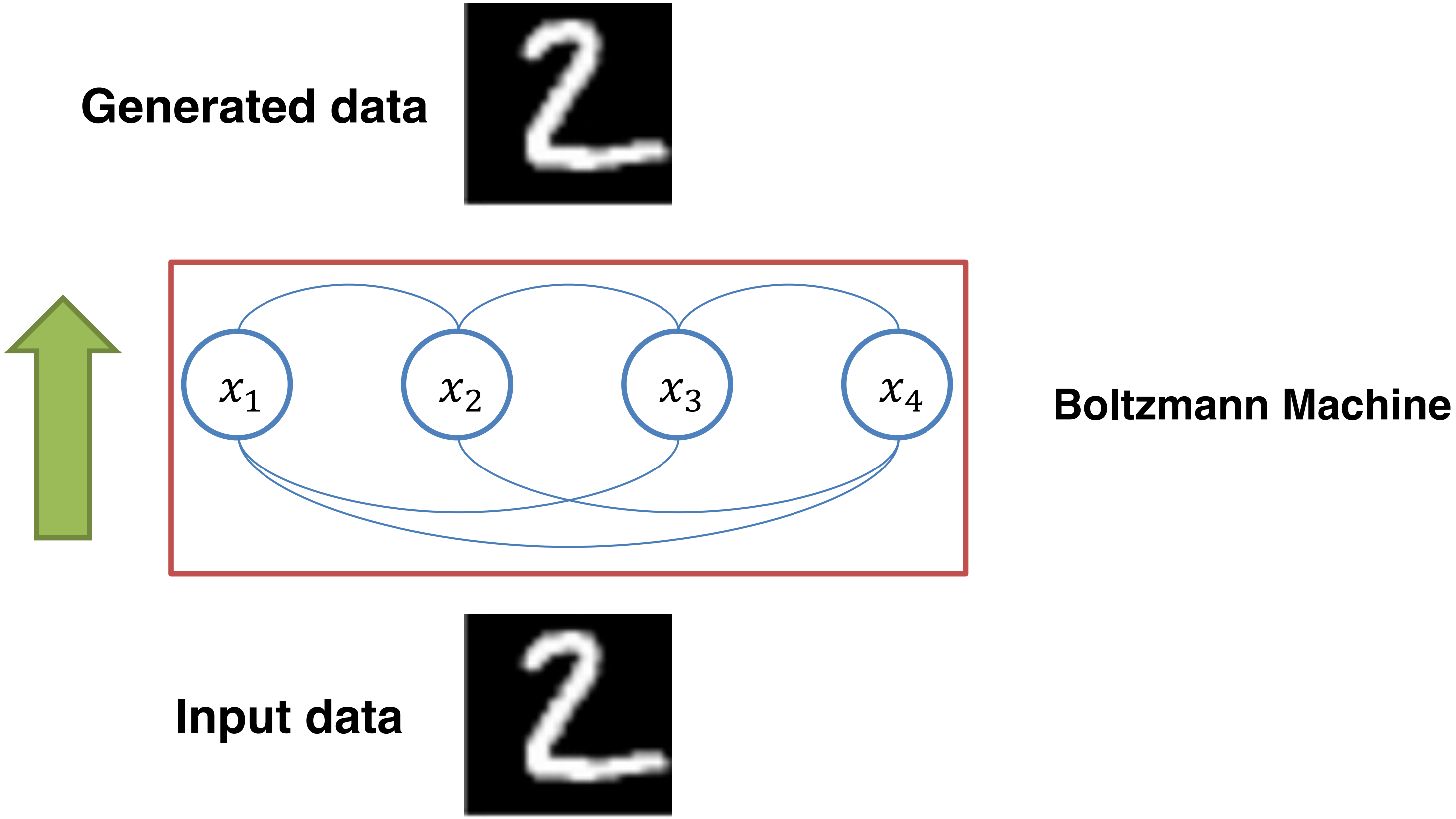


$$\hat{\boldsymbol{x}} = \boldsymbol{g}(\boldsymbol{y}\boldsymbol{W}' + \boldsymbol{b})$$

Boltzmann Machines

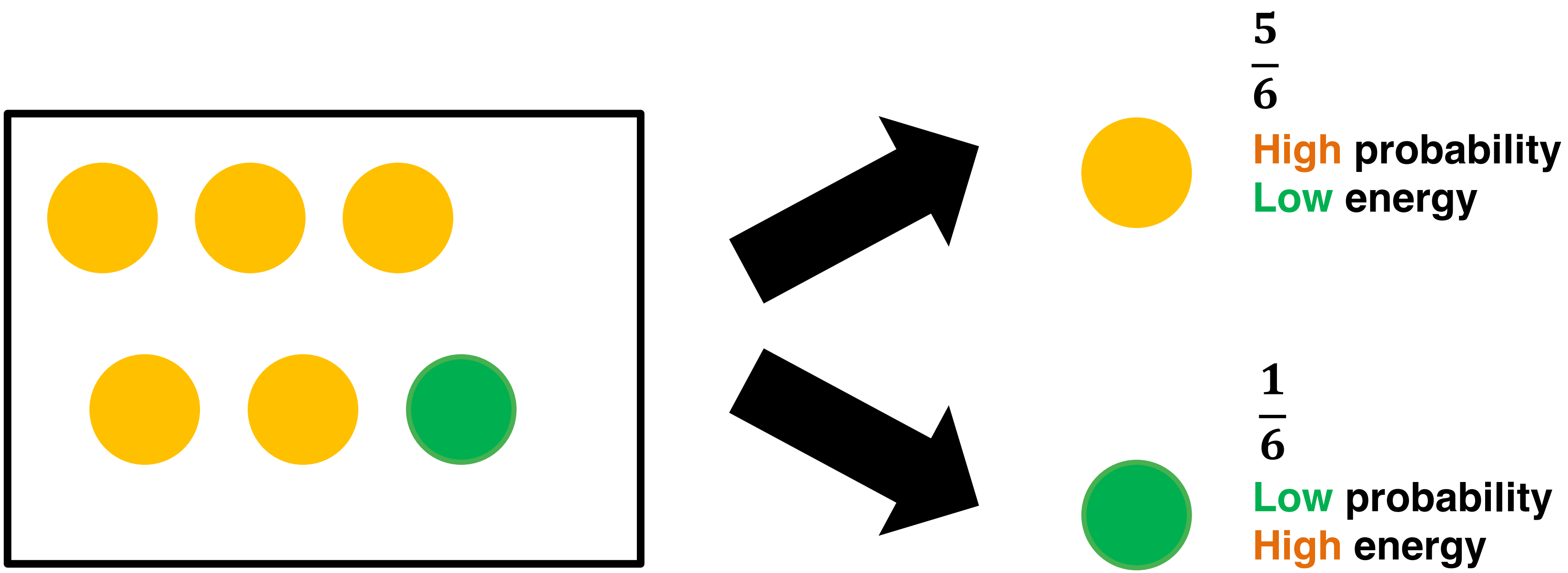
Boltzmann Machines (BM)

- BM is undirected graphical model and energy based model
- Each node is connected with all of other nodes



Energy Based Model

- **The meaning of energy**
  - a measure of the amount of information
  - an example of high energy event – winning a lottery
  - an example of low energy event – not winning a lottery

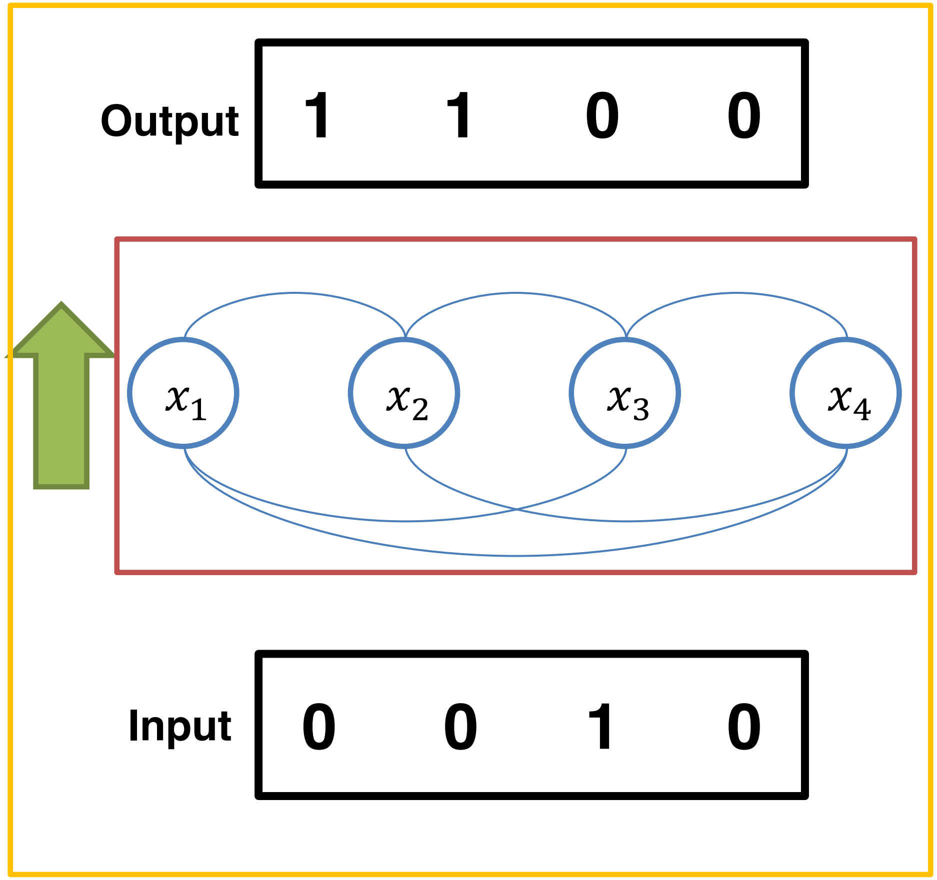


Energy Based Model

- The goal is to **lower the energy** and **increase the probability** that the generated data actually exists

$$E(x) = -\sum_i b_i x_i - \sum_i \sum_j w_{ij} x_i x_j$$

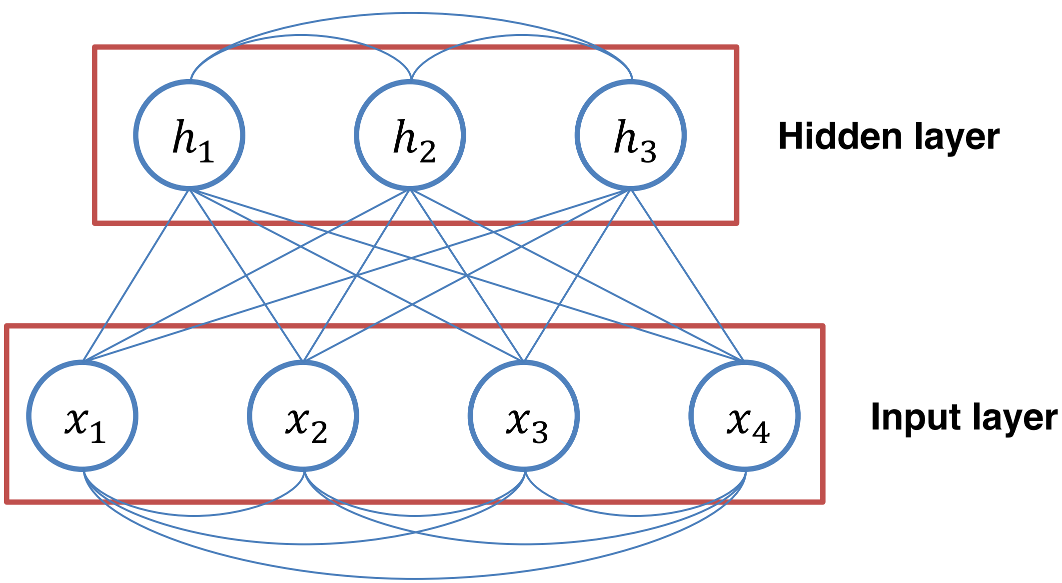
$$P(x) = \frac{1}{Z} exp(-E(x))$$



E : $energy$
U : $weight$
b : $bais$
Z : $partition\ function$

Multi Layer Boltzmann Machines

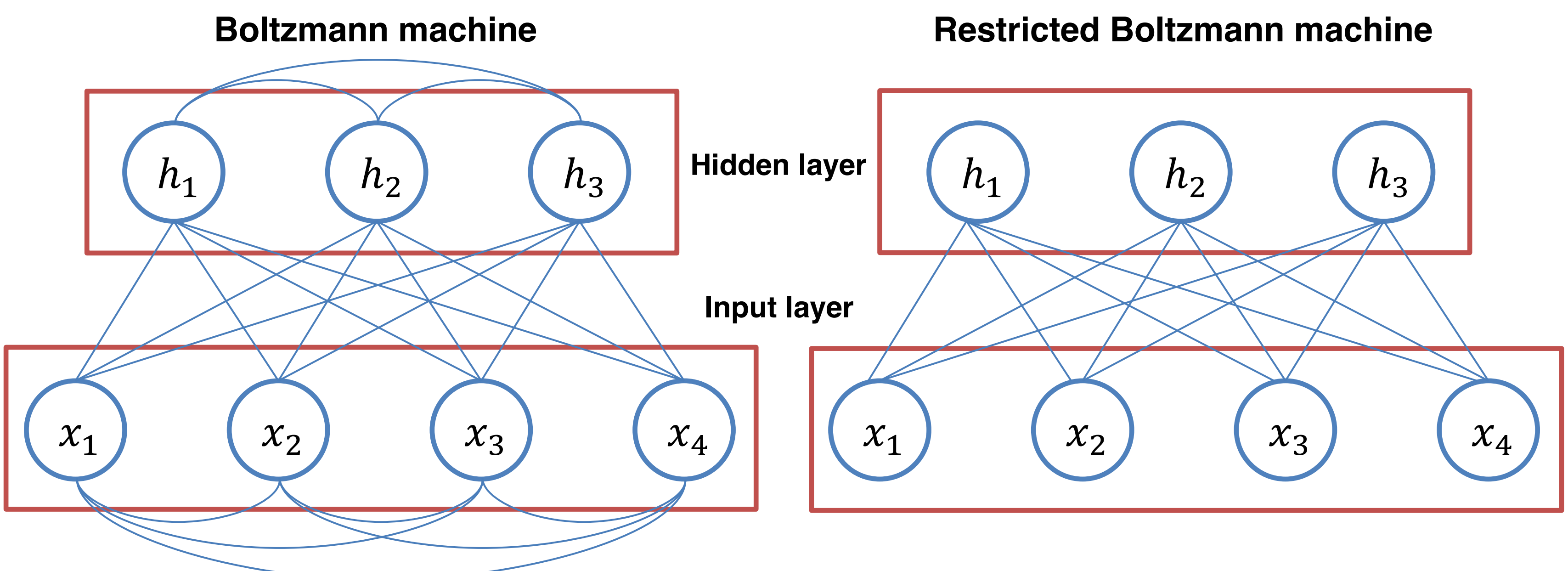
- Single layer BM can only generate linear data
- Multi layer BM can generate data with more complex distribution



Restricted Boltzmann Machines

Restricted Boltzmann Machines (RBM)

- **Characteristics of RBM**
  - no connection among same layer
  - training procedure is much simpler than BM

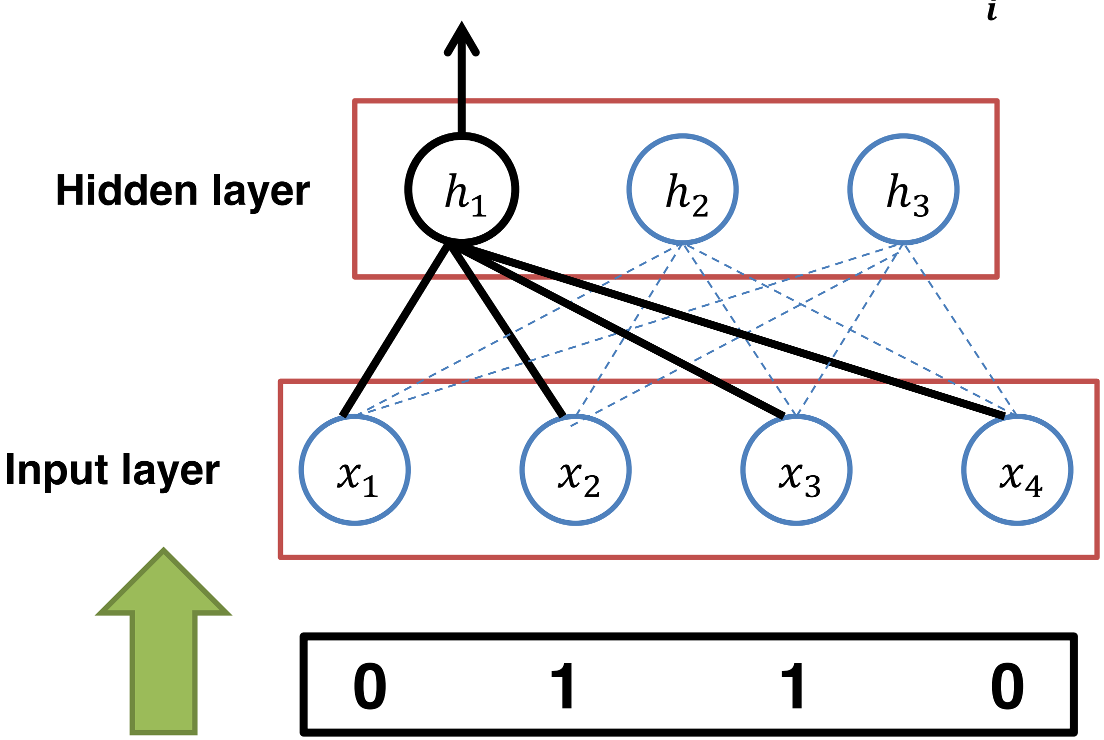


Data Generation of RBM

a : $bias\ of\ input\ layer$
b : $bias\ of\ hidden\ layer$
w : $weight$
$sigm$: $sigmoid\ function$

- **Calculate latent variables**

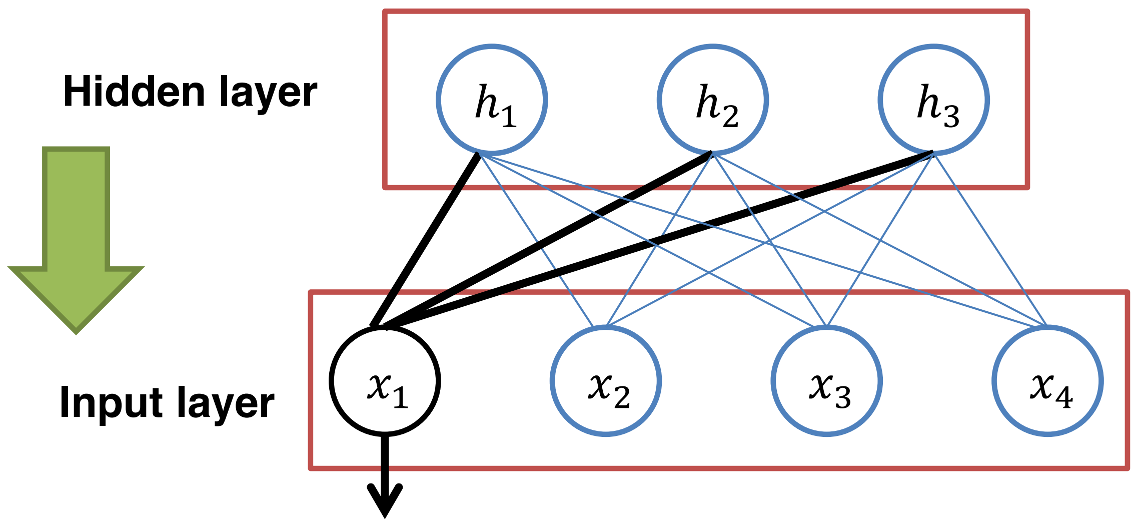
$$p(h_j = 1|x, \theta) = sigm(b_j + \sum_i w_{ij}x_i)$$



Data Generation of RBM

- **Calculate reconstruction value**

a : *bias of input layer*
b : *bias of hidden layer*
w : *weight*
sigm : *sigmoid function*



0.3	0.5	0.9	0.1

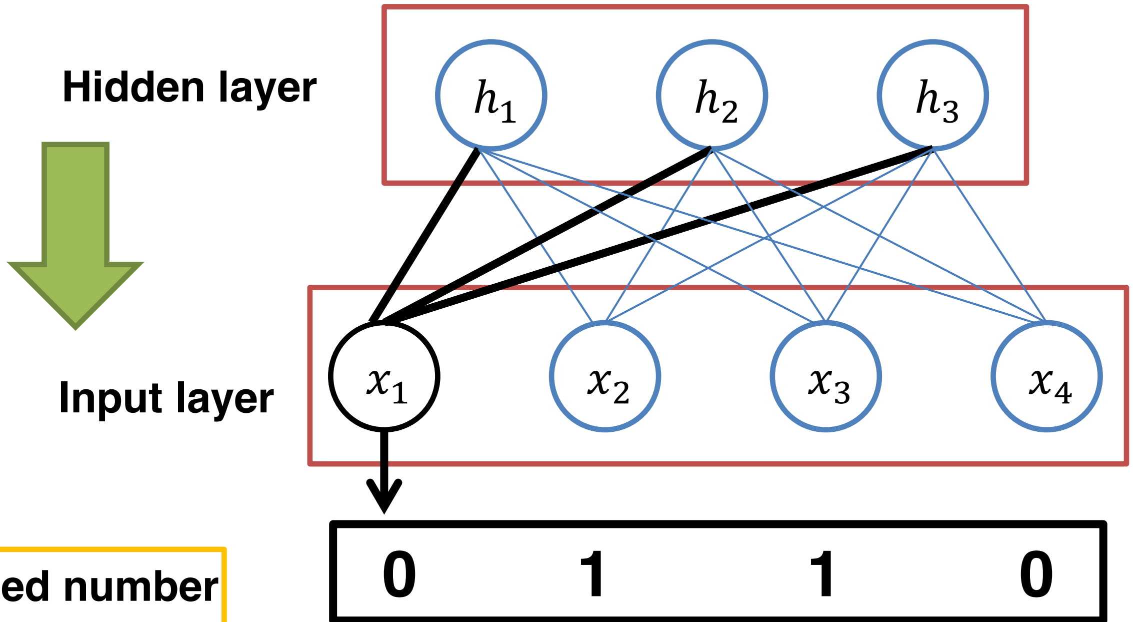
$$p(x_i = 1|h,\theta) = sigm(a_j + \sum_j w_{ij} h_i)$$

randomly generated number
0.4

Data Generation of RBM

a : *bias of input layer*
b : *bias of hidden layer*
w : *weight*
sigm : *sigmoid function*

- **Calculate reconstruction value**



randomly generated number
0.4

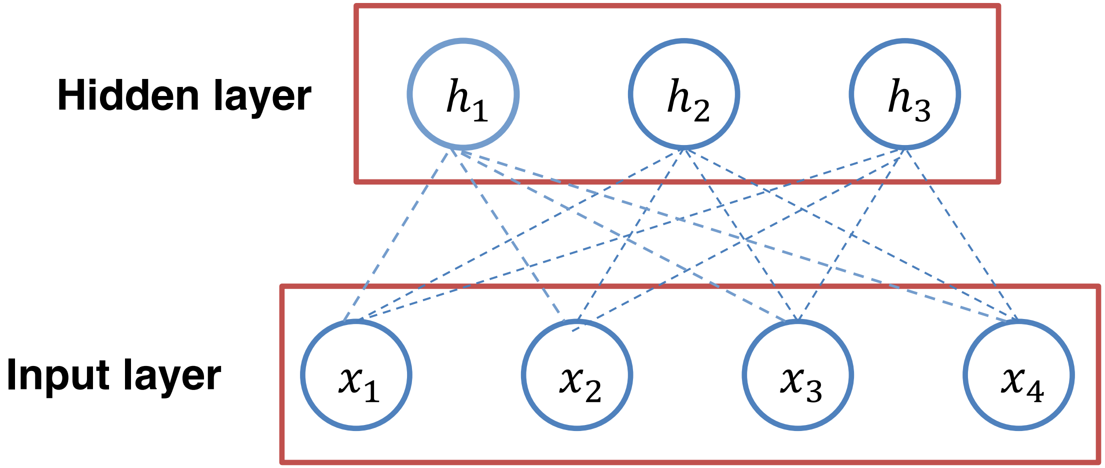
$$p(x_i = 1|h, \theta) = sigm(a_j + \sum_j w_{ij} h_i)$$

Probability of RBM

a : bias of input layer
b : bias of hidden layer
E : energy function
Z : partition function

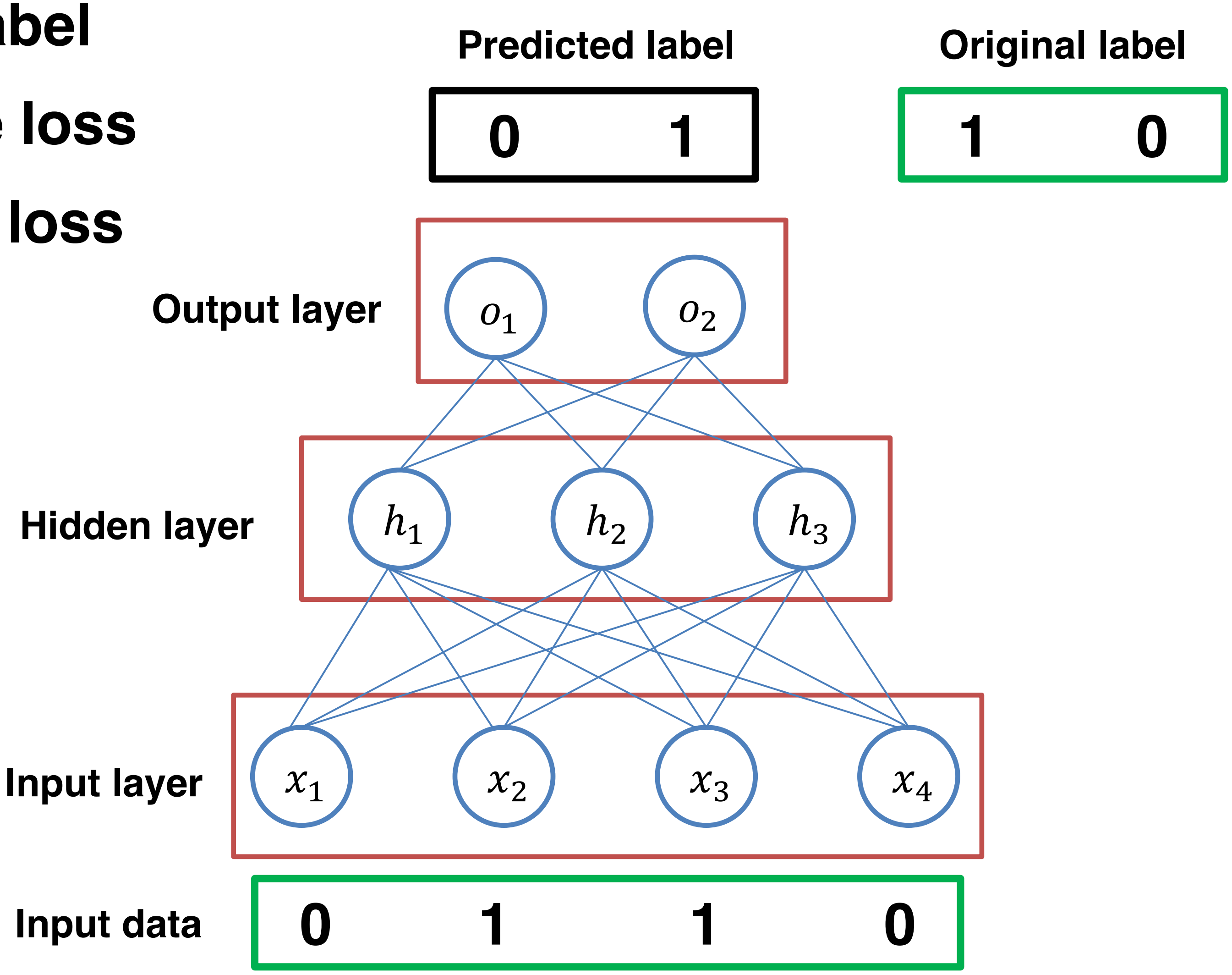
$$E = -\sum_i a_i v_i - \sum_j b_j h_j - \sum_i \sum_j w_{ij} v_i h_j$$

$$P(x) = \frac{1}{Z} exp(-E(x))$$



Training of Multi Layer Perceptron (MLP)

1. Predict label
2. Calculate loss
3. Minimize loss



Training of Multi Layer Perceptron (MLP)

1. Predict label
2. Calculate loss
3. Minimize loss

α : $learning\ rate$
Δw : $gradient$

- Parameter update with gradient decent

$$\Delta w_{ij} = \frac{\delta loss}{\delta w_{ij}}$$

$$w_{ij} \leftarrow w_{ij} - \alpha \cdot \Delta w_{ij}$$

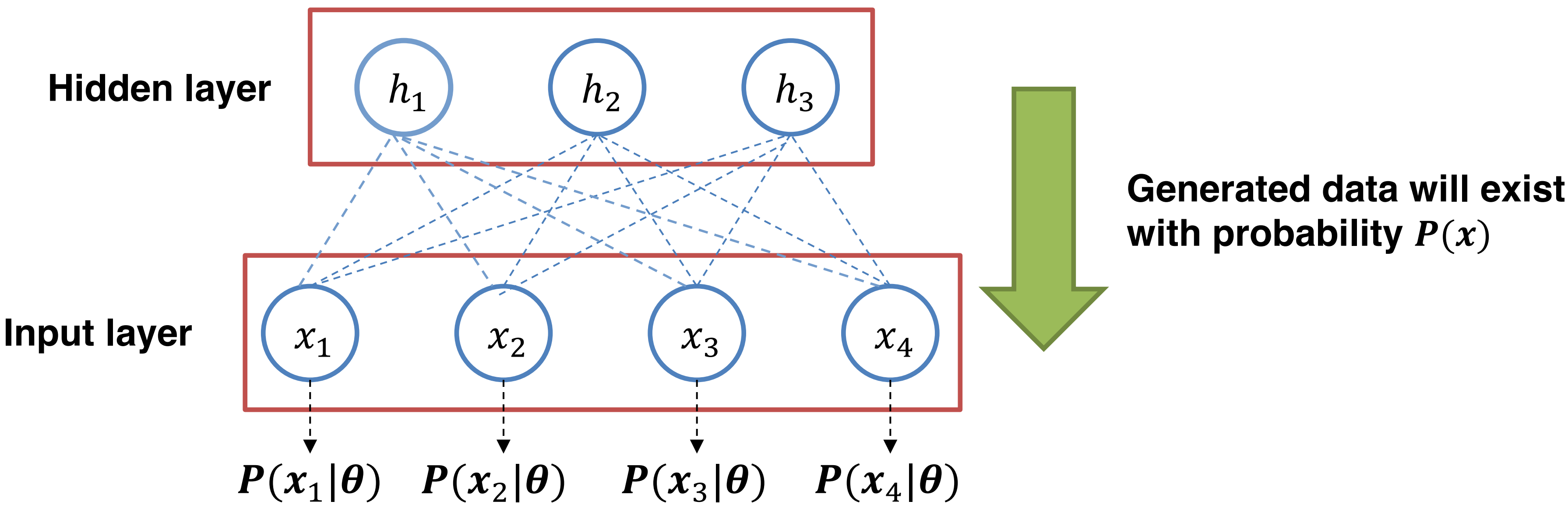
Likelihood of RBM

L : *likelihood*
E : *energy function*
Z : *partition function*

- **RBM learns distribution of data by maximizing likelihood**

$$P(x) = \frac{1}{Z} exp(-E(x))$$

$$L(\theta) = \prod_{n=1}^{N} P(x_n|\theta)$$



Training of RBM

α : *learning rate*
L : *likelihood*
E : *energy function*
Z : *partition function*

- **Probability**

Difficult to calculate

$$P(x) = \boxed{\frac{1}{Z}} exp(-E(x))$$

- **Likelihood**

$$L(\theta) = \prod_{n=1}^{N} P(x_n|\theta)$$

- **Log likelihood**

$$log(L(\theta)) = \sum_{n=1}^{N} log(P(x_n|\theta)) = \sum_{n=1}^{N} \{-E - log(Z)\}$$

Differentiate

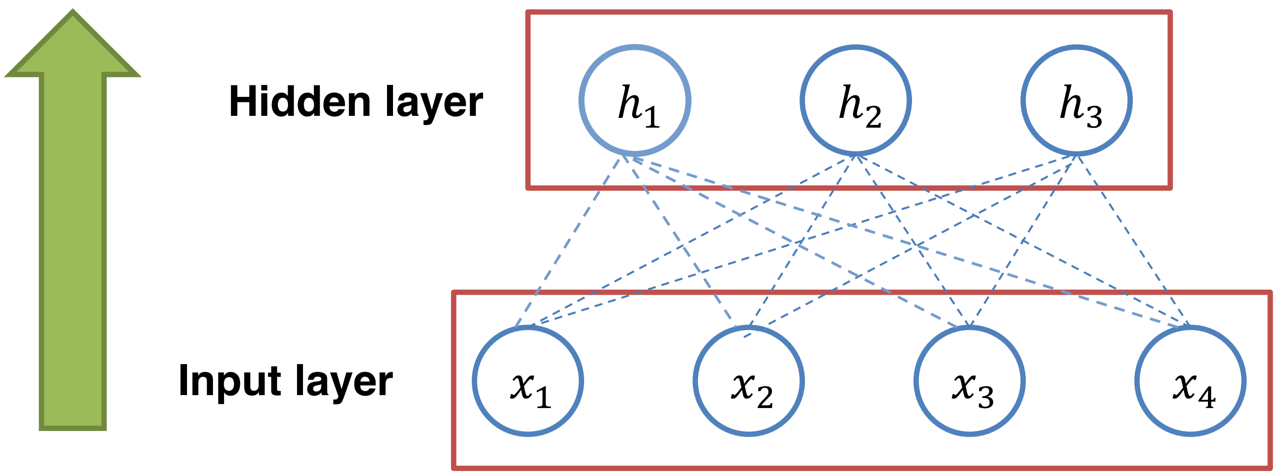
- **Weight update formula**

$$w_{ij} \leftarrow w_{ij} + \alpha \frac{\delta log(L(\theta))}{\delta w_{ij}}$$

Contrastive Divergence (CD)

- Approximate algorithm of training of RBM

h^0 | 0.2 | 0.7 | 0.6



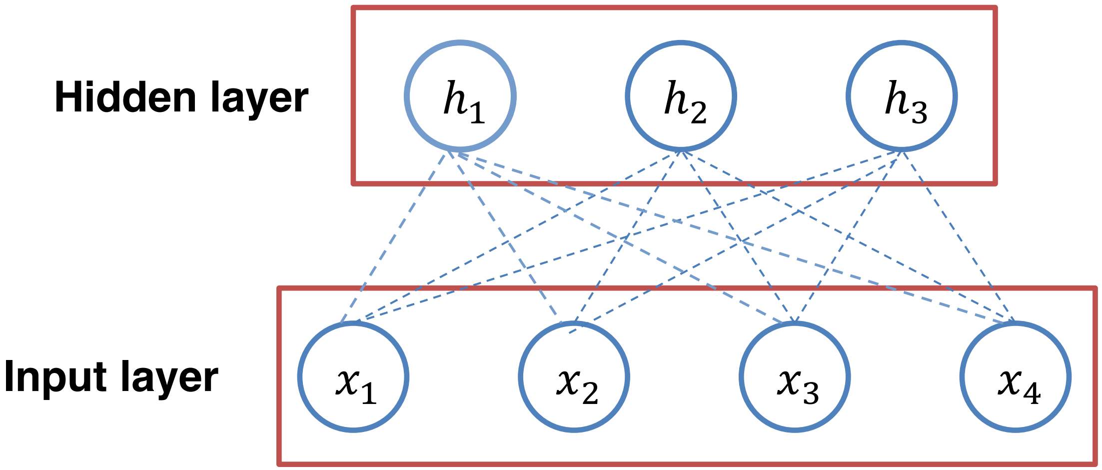
x^0 | 0 | 1 | 1 | 0

Contrastive Divergence (CD)

- **Approximate algorithm of training of RBM**

h^0

0.2	0.7	0.6



x^0

0	1	1	0

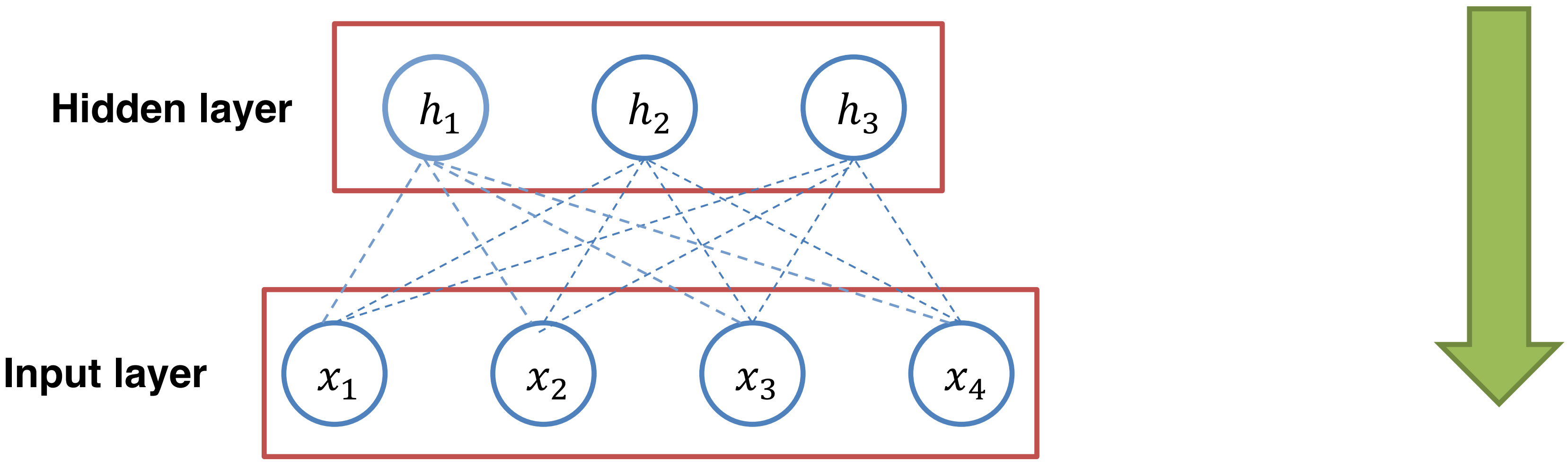
p^1

0.1	0.5	0.9	0.3

Contrastive Divergence (CD)

- **Approximate algorithm of training of RBM**

h^0 | 0.2 | 0.7 | 0.6 |



x^0 | 0 | 1 | 1 | 0 |

x^1 | 0 | 1 | 1 | 0 |

Contrastive Divergence (CD)

- **Approximate algorithm of training of RBM**

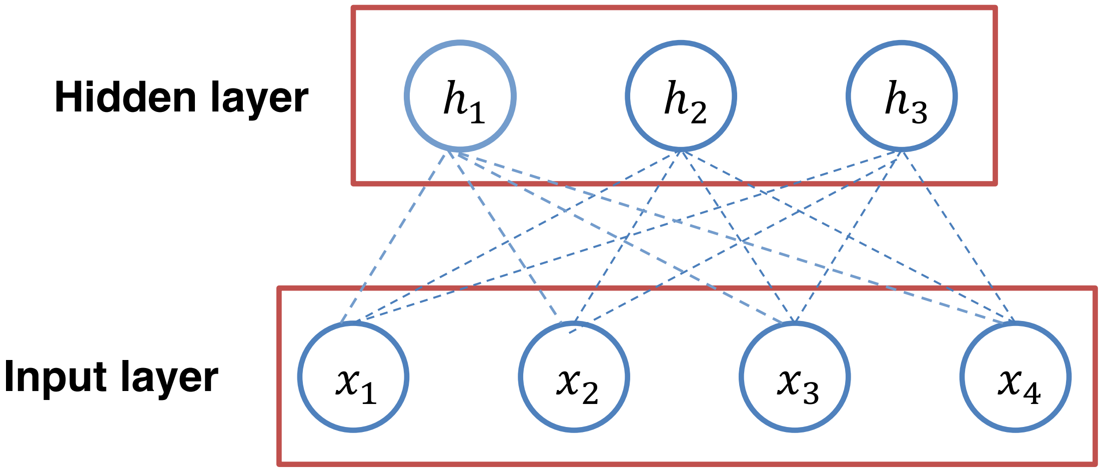
Iterate k time

h^0

0.2	0.7	0.6

h^1

0.8	0.2	0.3



x^0

0	1	1	0

x^1

0	1	1	0

Weight Update with CD

- **Gradient of weight**

$$\Delta w_{ij} = x_i^0 h_j^0 - x_i^k h_j^k$$

- **Gradient of bias of input layer**

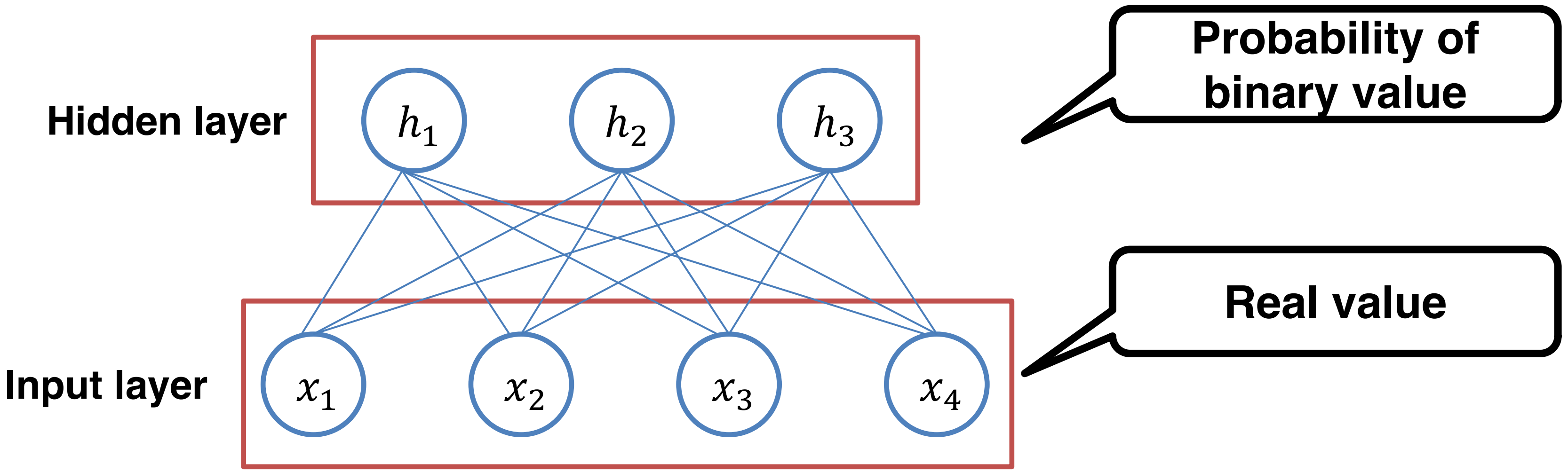
$$\Delta a_i = x_i^0 - x_i^k$$

- **Gradient of bias of hidden layer**

$$\Delta b_j = h_j^0 - h_j^k$$

Boltzmann Machines for Real-Valued Data

- **Gaussian-Bernoulli RBM**



Boltzmann Machines for Real-Valued Data

σ : *standard deviation*

- Energy function

$$E = -\sum_i \frac{(v_i - a_i)^2}{2\sigma_i^2} - \sum_j b_j h_j - \sum_i \sum_j w_{ij} \frac{v_i}{\sigma_i} h_j$$

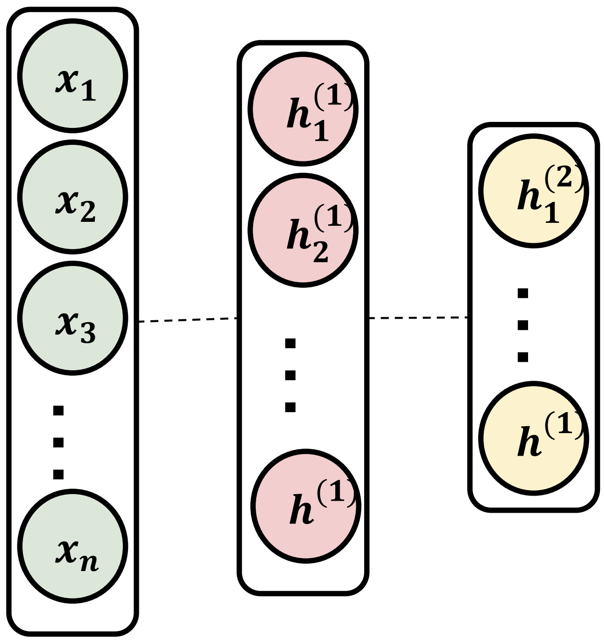
- Probability

$$P(x_i|h) \propto exp\left(-\frac{(v_i - a_i - \sum_j w_{ij} h_j)^2}{2\sigma_i^2}\right)$$

Deep Belief Networks

Deep Belief Networks (DBN)

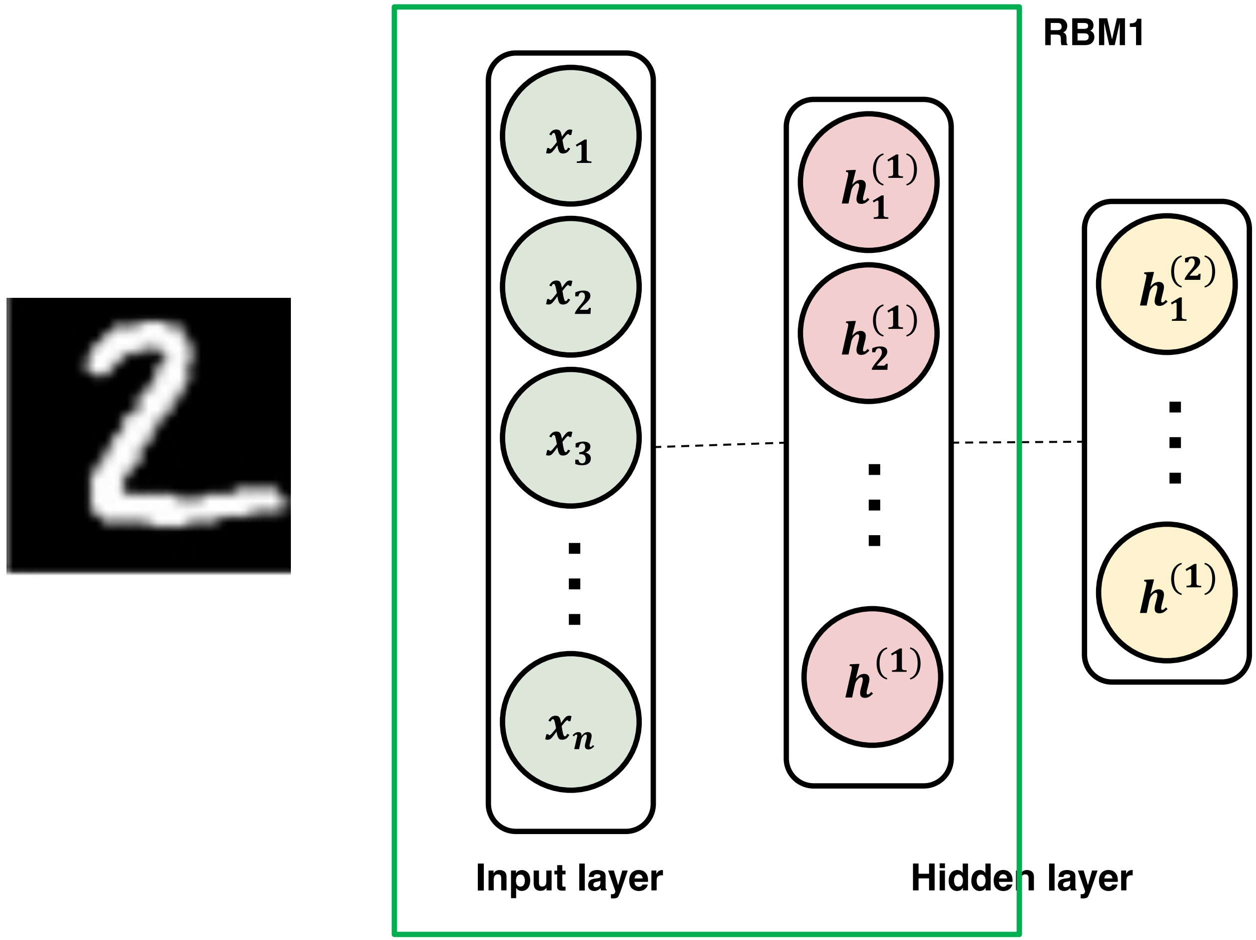
- **The first non-convolutional models to successfully admit training of deep architectures (Hinton et al, 2006)**



Input layer Hidden layer

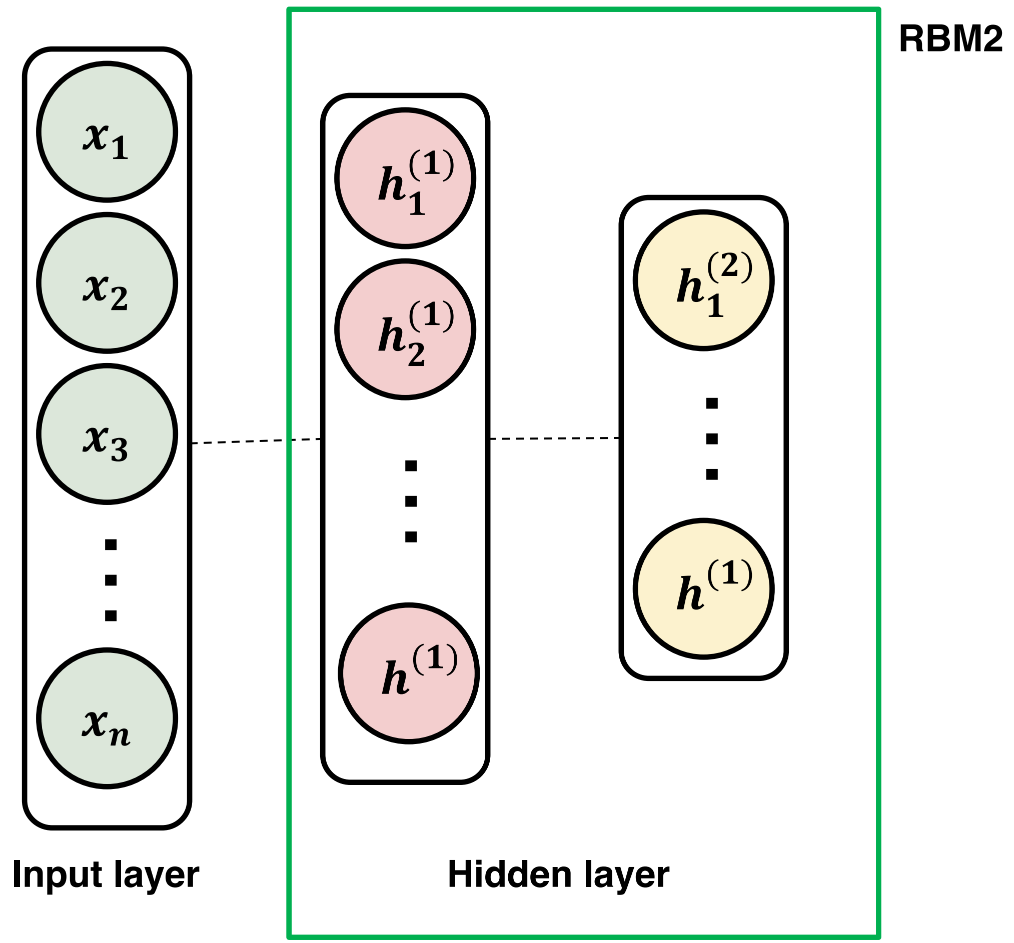
DBN for Classification

- **Layer-wise pre-training (unsupervised)**



DBN for Classification

- **Layer-wise pre-training (unsupervised)**



DBN for Classification

- **Fine tuning (supervised)**