Unsupervised Learning

Hung-yi Lee

Introduction

- We already learn some machine learning techniques.
- With labelled data, you can do any thing (hopefully).
- Labelling data is expensive.
- What can we do if there is no sufficient training data?
- Unsupervised Learning Approaches
  - Restricted Boltzmann Machine (RBM)
  - Auto-encoder

Semi-supervised Learning

Labelled data

cat

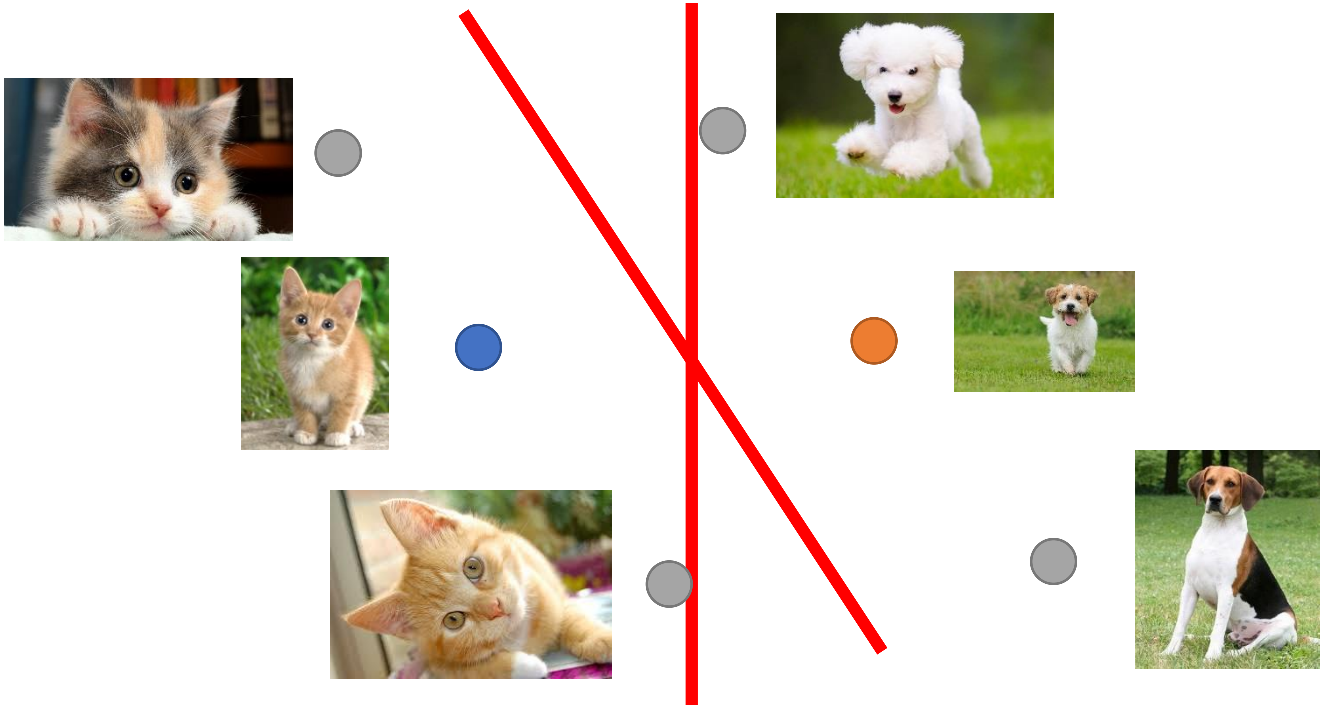
dog

Unlabeled data

(Image of cats and dogs without labeling)

Semi-supervised Learning

- Why semi-supervised learning helps?



The distribution of the unlabeled data tell us something.

Transfer Learning

Labelled data

cat

dog

Labeled data

elephant

elephant

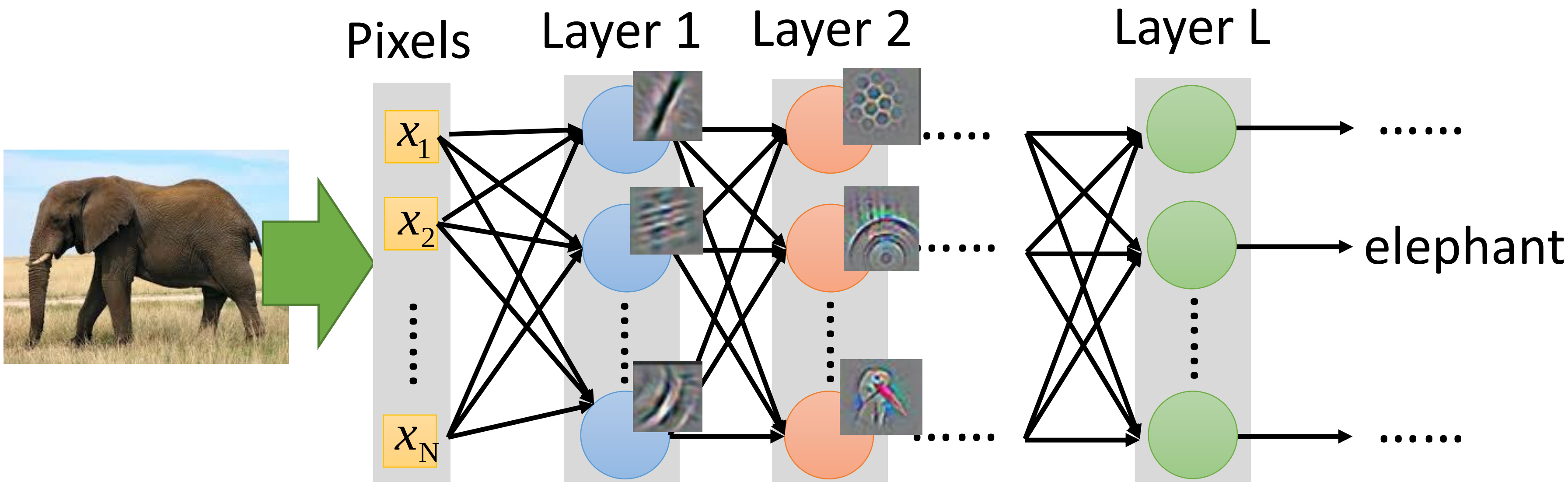
tiger

tiger

Not related to the task considered

Transfer Learning

- Widely used on image processing
  - Using sufficient labeled data to learn a CNN
  - Using this CNN as feature extractor



Transfer Learning

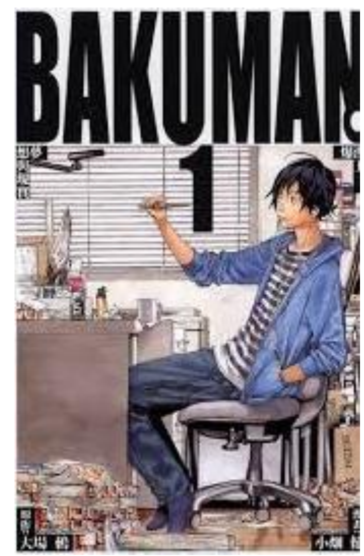
- Example in real life

<u>研究生 online</u>

<u>漫畫家 on-line</u>

研究生生存守則

研究生生存守則		
研究生	←	漫畫家
指導教授	←	責編
跑實驗	←	畫分鏡
投稿期刊	←	投稿 jump



爆漫王

Self-taught Learning

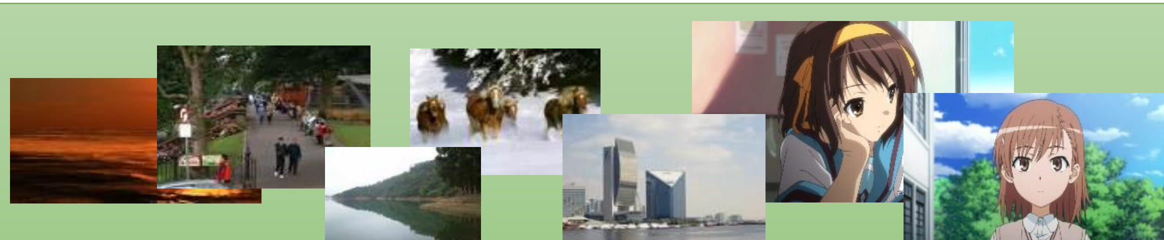
- Transfer learning with unlabeled data is not related to the task.

Labelled data

cat

dog

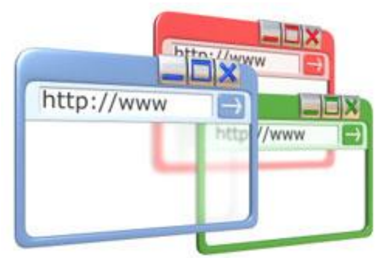
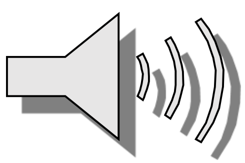
Unlabeled data



(Just crawl millions of images from the Internet)

Self-taught Learning

- Sometimes unlabeled data is not related to the task.

	Labelled data	Unlabeled data
Digit Recognition	 Digits	 character
Document Classification	 News	 Webpages
Speech Recognition	 Taiwanese	YouTube English Chinese ……

Why self-taught learning can work?

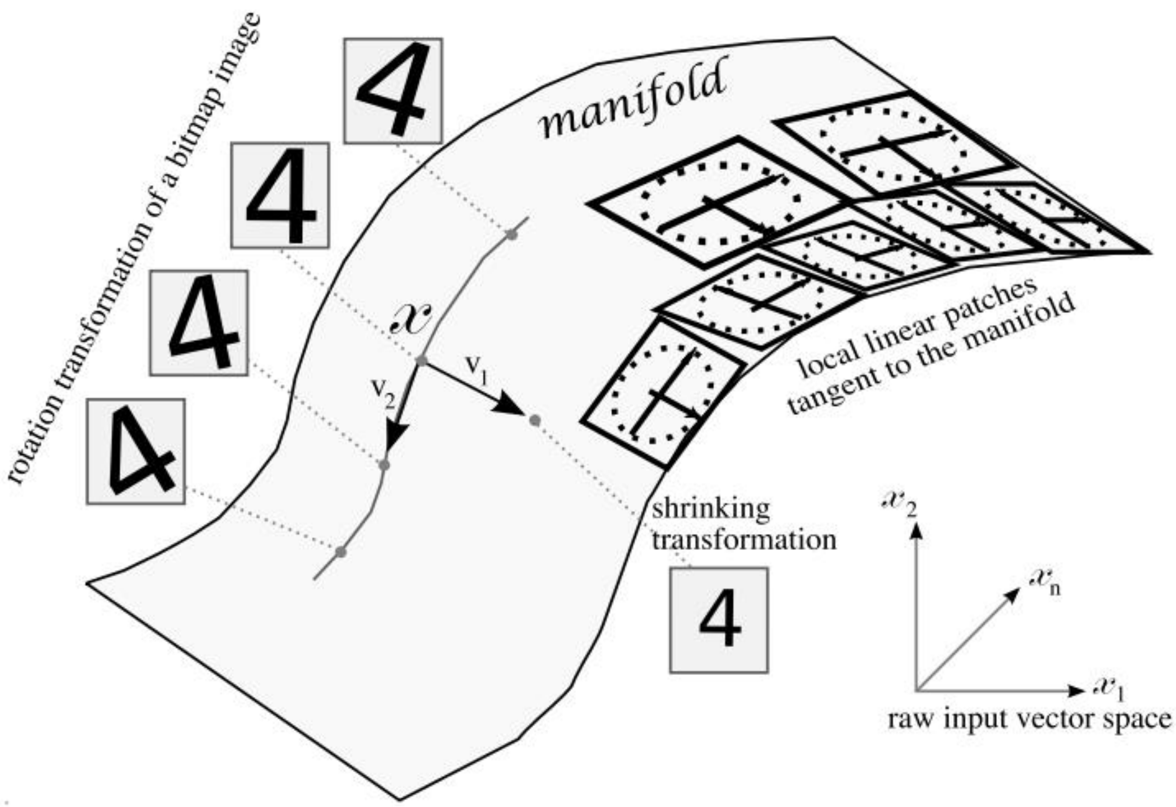
- Why Unlabeled and unrelated data can help?
- Find the latent factors controlling the observation

observation

Latent factor

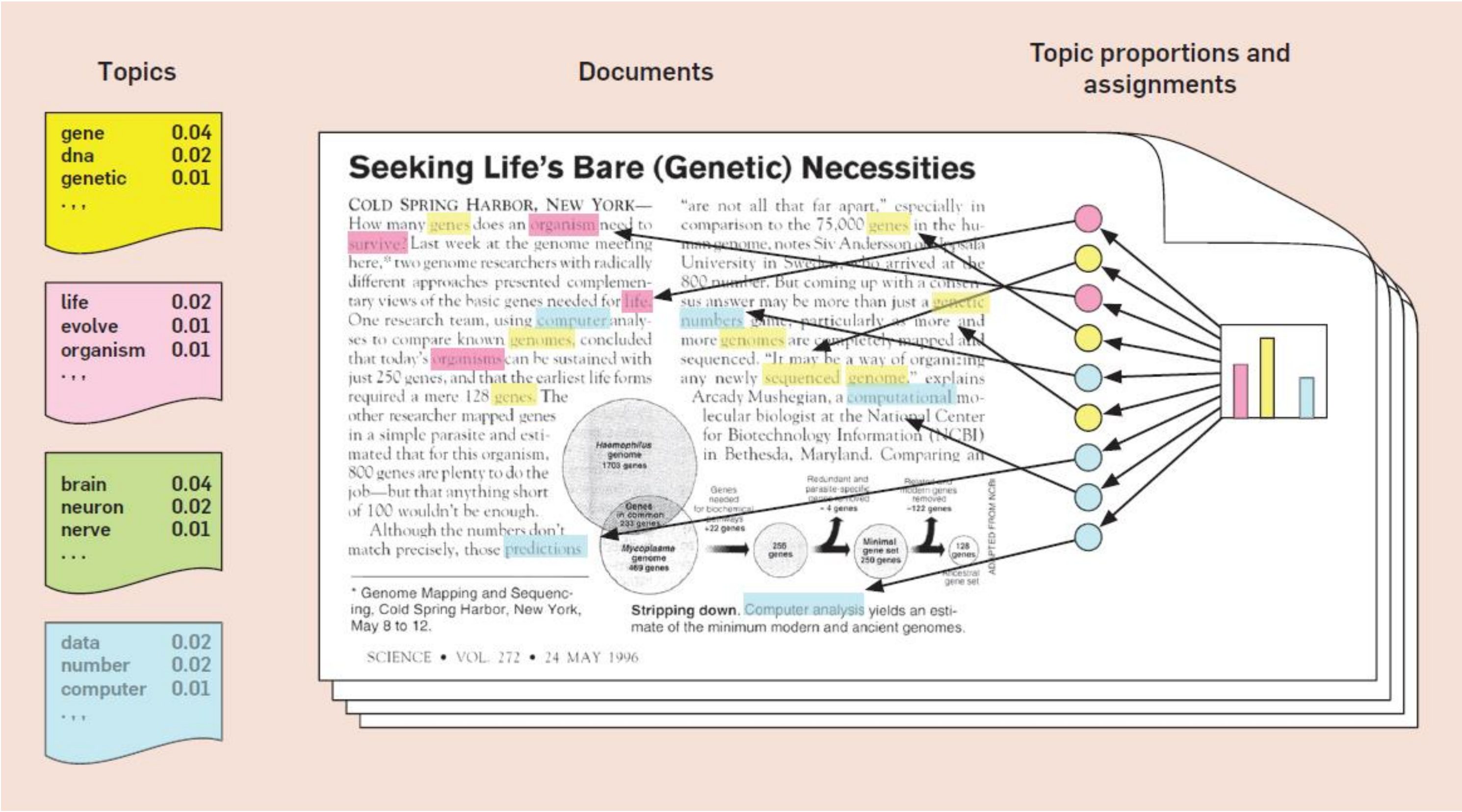
Latent Factors

- Handwritten Digits



Latent Factors

- Documents

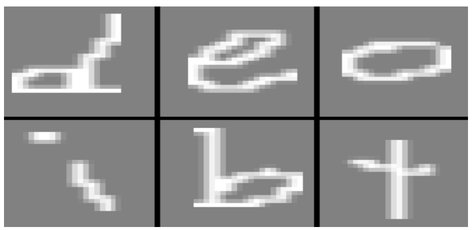


http://deliveryimages.acm.org/10.1145/2140000/2133826/figs/f1.jpg

Recommendation System

How to exploit latent factors

- Handwritten Digits



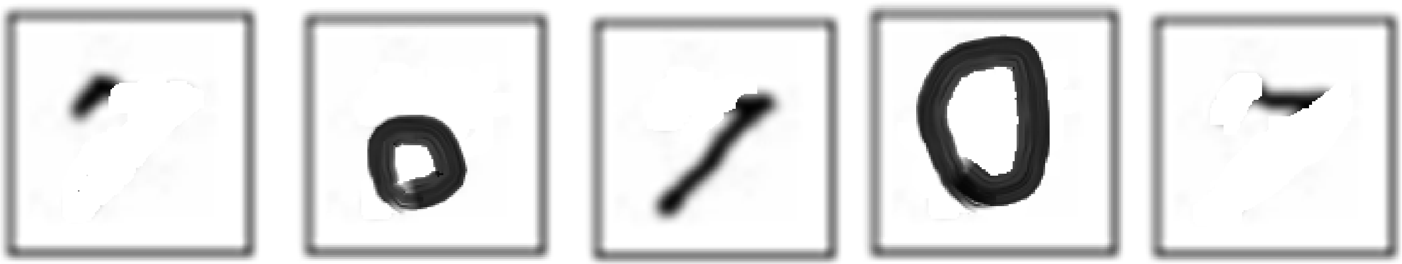
The hand written images are composed of strokes.

<u>Strokes (Latent Factors)</u>

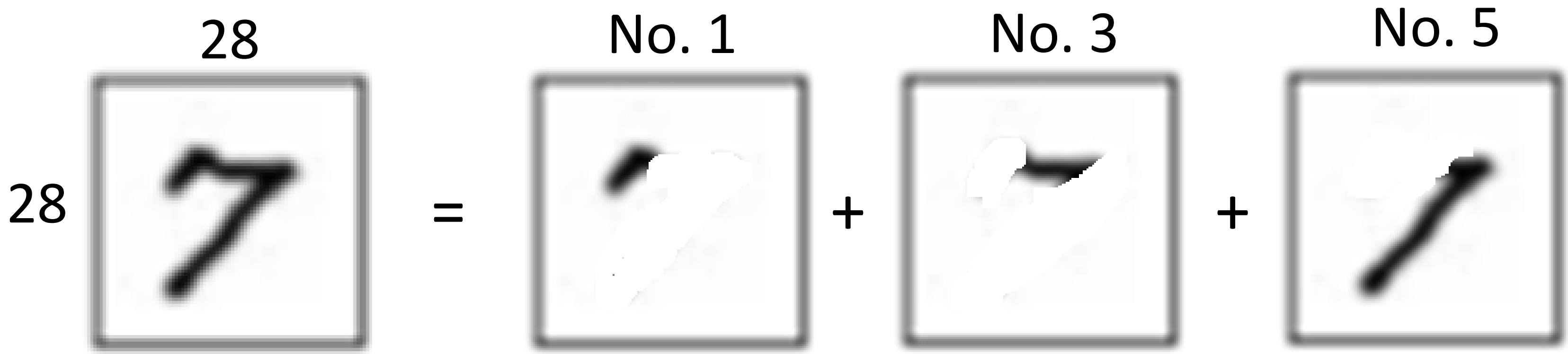
No. 1 No. 2 No. 3 No. 4 No. 5 ....

How to exploit latent factors

<u>Strokes (Latent Factors)</u>



No. 1 No. 2 No. 3 No. 4 No. 5 ....



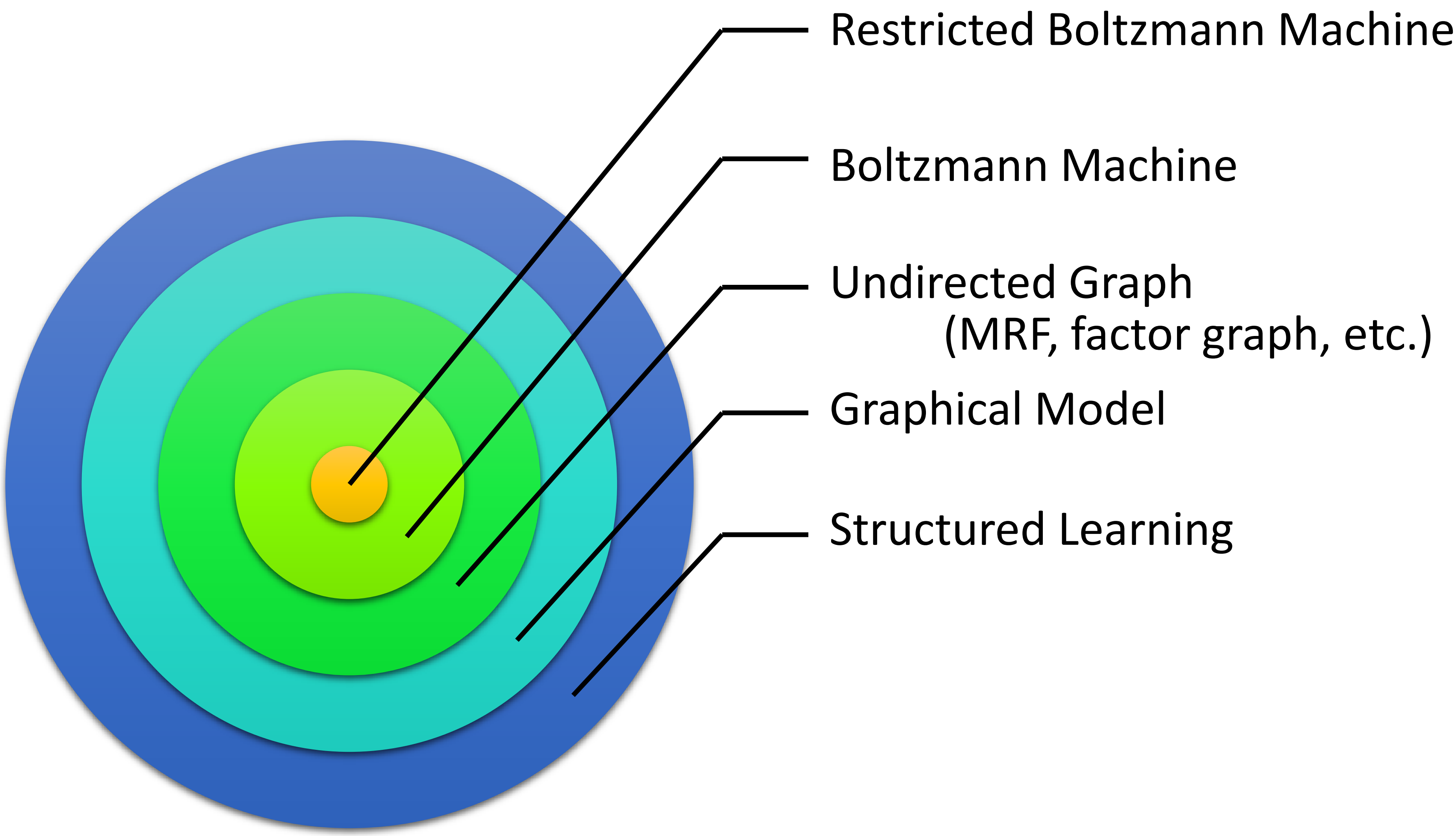
Represented by
28 X 28 = 784 pixels

[1 0 1 0 1 0 ....]

(simpler representation)

Restricted Boltzmann Machine (RBM)

Where are we?


Restricted Boltzmann Machine
Boltzmann Machine
Undirected Graph
(MRF, factor graph, etc.)
Graphical Model
Structured Learning

Boltzmann Machine

- There is a set of variables S = {s_1, s_2, …, s_i, … , s_K}
- The values of each variable $s_i \in \{0,1\}$
  - will be generalized later

<u>Evaluation function</u>

$$E(S) = \sum_i a_i s_i + \sum_{i<j} w_{ij} s_i s_j$$

When $s_i = 1$, the evaluation function gains a_i

When $s_i = 1$ and $s_j = 1$, the evaluation function gains w_{ij}

a_i and w_{ij} are learned from data.

Boltzmann Machine - Example

$s = \{s_1, s_2, s_3, s_4\}$

$E(s_1 = 1, s_2 = 0, s_3 = 0, s_4 = 1) = a_1 + a_4 + w_{14}$

$E(s_1 = 0, s_2 = 0, s_3 = 0, s_4 = 0) = 0$

$$E(s_1 = 1, s_2 = 1, s_3 = 1, s_4 = 0) = a_1 + a_2 + a_3 + w_{12} + w_{13} + w_{23}$$

If $a_i > 0$, s_i is likely to be 1

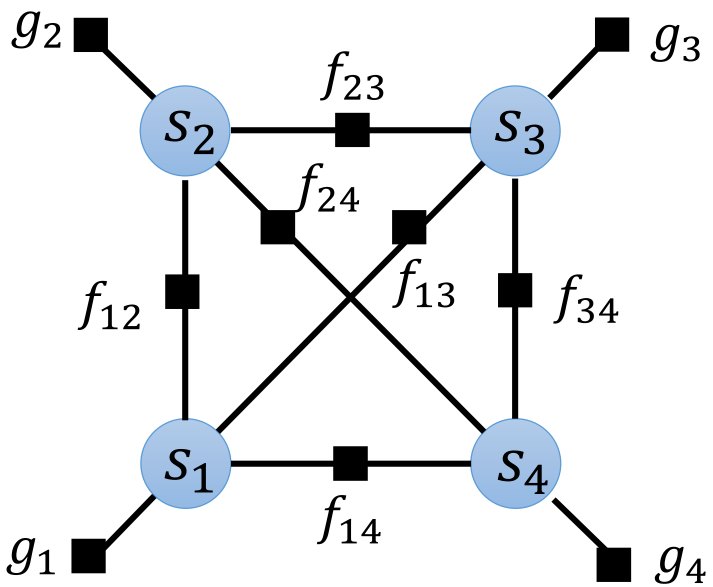
If $w_{ij} > 0$, s_i and s_j are tend to be 1 together

Probability Point of View

$$P(S) = \frac{e^{E(S)}}{\sum_{S'} e^{E(S)}}$$

Boltzmann Machine
- Factor Graph

- There are factors for each node and factors between each node pair



Factor for a node:

$$g_i(s_i) = \begin{cases} a_i & s_i = 1 \\ 0 & else \end{cases}$$

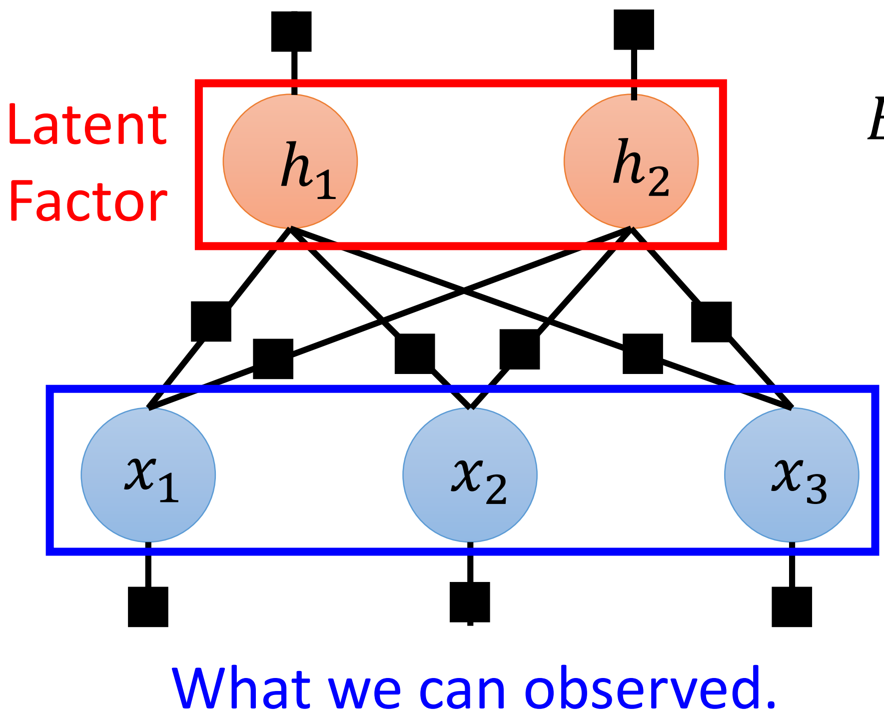
Factor between nodes:

$$f_{ij}(s_i, s_j) = \begin{cases} w_{ij} & s_i = 1, s_j = 1 \\ 0 & else \end{cases}$$

$$E(S) = \sum_i g_i(s_i) + \sum_{i<j} f_{ij}(s_i, s_j) = \sum_i a_i s_i + \sum_{i<j} w_{ij} s_i s_j$$

Restricted Boltzmann Machine (RBM)

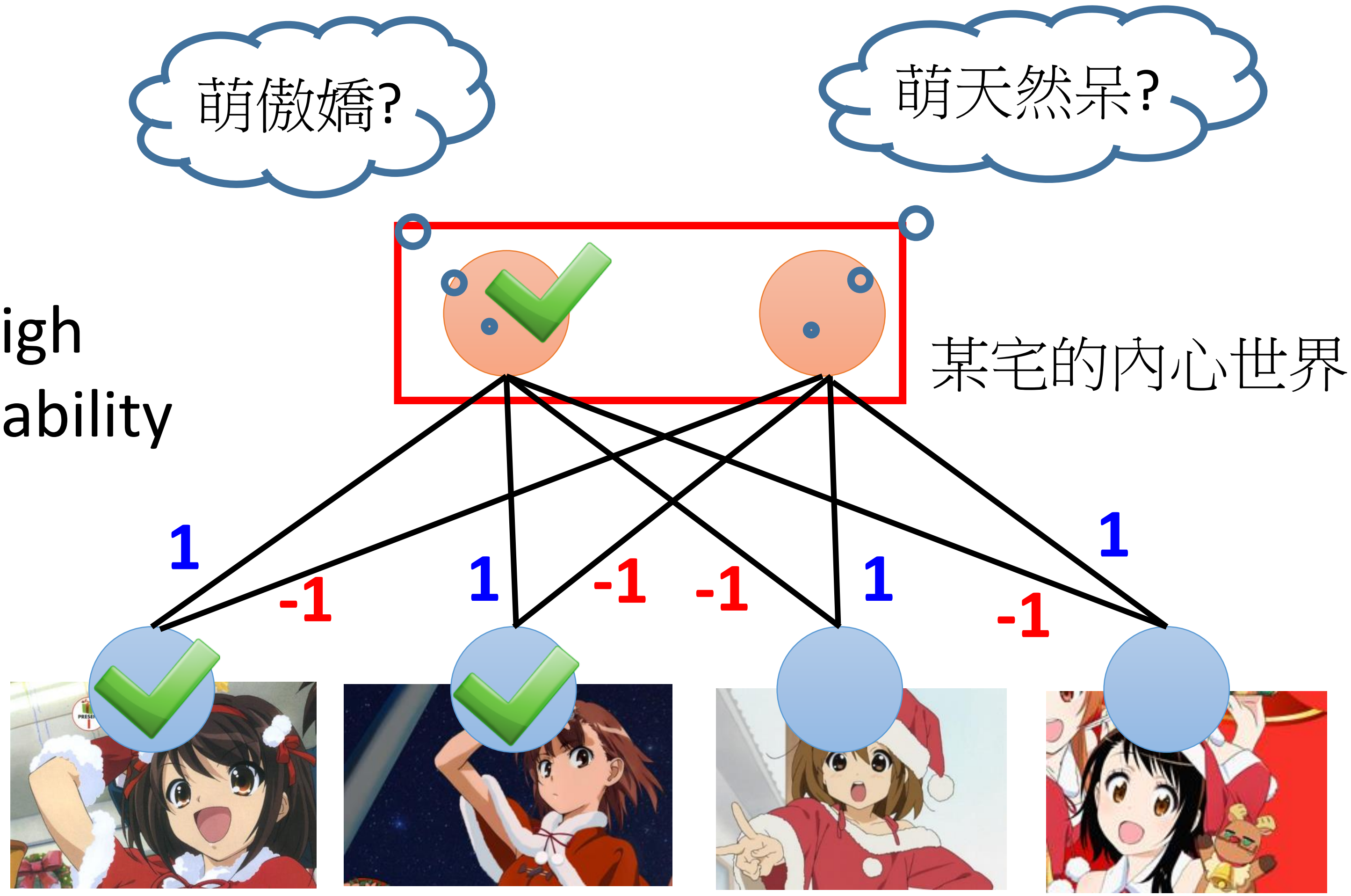
- The variables are separated into two sets
- The variables in the same sets are not connected



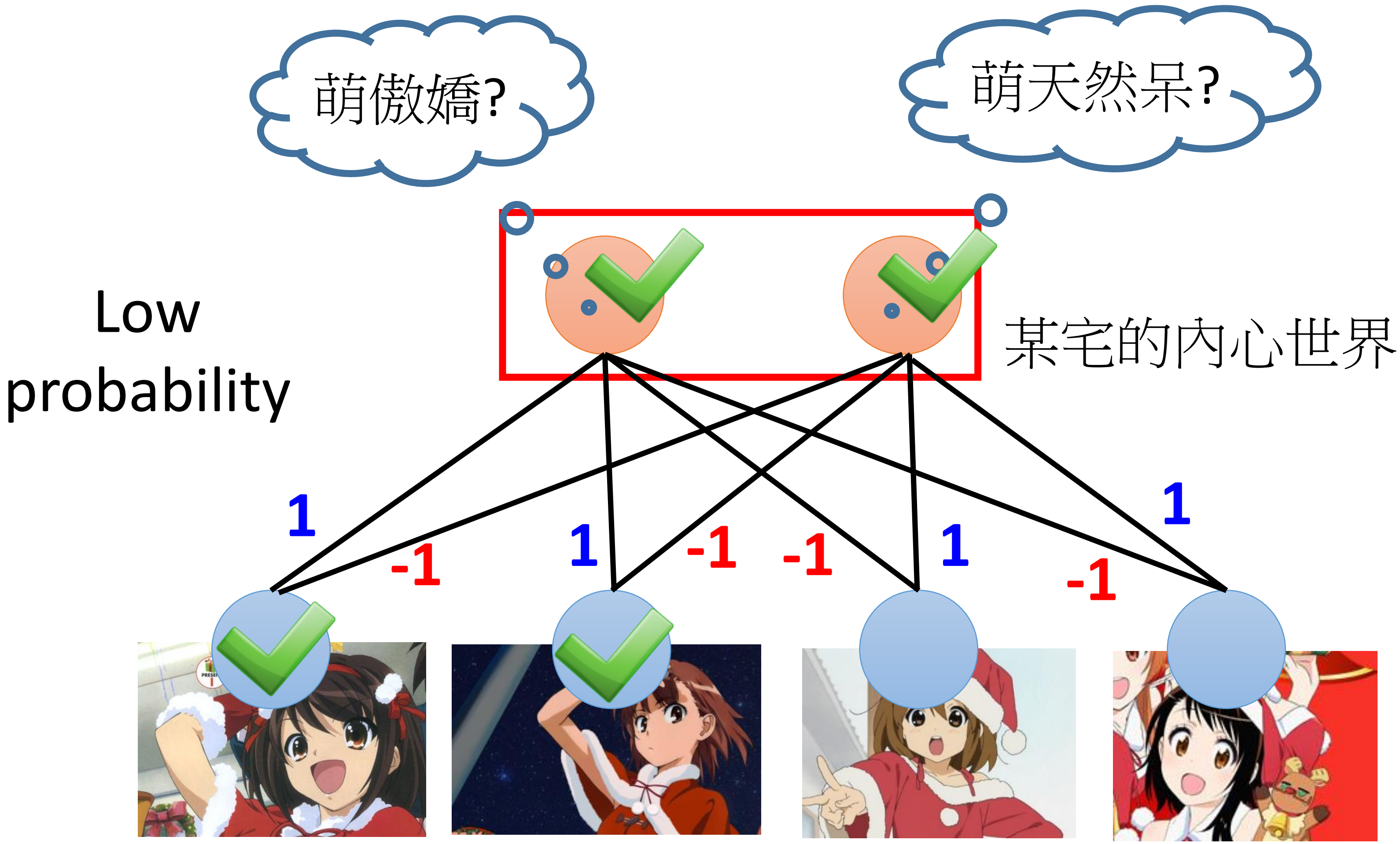
$$E(x,h) = \sum_{h_i} b_i h_i + \sum_{x_j} c_j x_j + \sum_{h_i,x_j} w_{ij} h_i x_j$$

RBM – Example

High probability

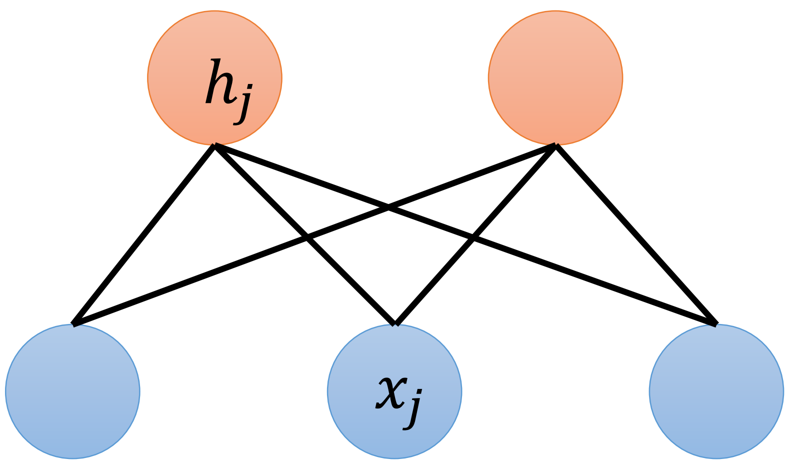


RBM – Example



RBM - Inference

- Given the parameters, compute $P(h_j = 1|x)$ is simple



Given x, all h_j are independent

Given h, all x_i are independent

(see the reference)

RBM

$$E(x,h)=\sum_{h_i} b_i h_i + \sum_{x_j} c_j x_j + \sum_{h_i,x_j} w_{ij} h_i x_j$$

- Given the parameters, compute $P\left(h_j=1|x\right)$ is simple

$$P(h_i=1|x)=P(h_i=1|x,h_{-i})$$

$$P(x,h)=\frac{e^{E(x,h)}}{\sum_{x',h'} e^{E(x',h')}}$$

$$=\frac{P(x,h_{-i},h_i=1)}{P(x,h_{-i},h_i=1)+P(x,h_{-i},h_i=0)}$$

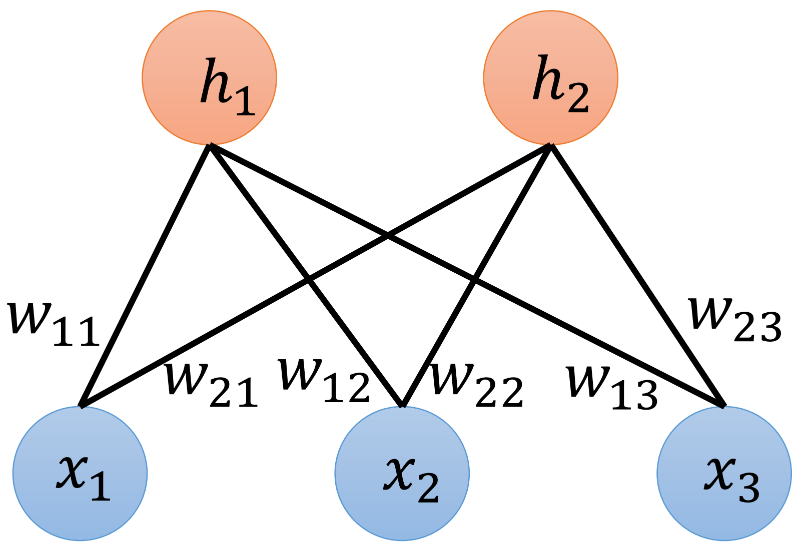
$$=\frac{e^{E(x,h_{-i},h_i=1)}}{e^{E(x,h_{-i},h_i=1)}+e^{E(x,h_{-i},h_i=0)}}$$

Only the terms related to h_i are different

$$=\frac{e^{b_i+\sum_{x_j} w_{ij}x_j}}{e^{b_i+\sum_{x_j} w_{ij}x_j}+1}=\frac{1}{1+e^{-\left(b_i+\sum_{x_j} w_{ij}x_j\right)}}=sig\left(b_i+\sum_{,x_j} w_{ij}x_j\right)$$

RBM - Inference

- Given the parameters, compute $P\left(h_j = 1|x\right)$ is simple



Given x_1, x_2 and x_3

$P(h_1 = 1|x)$

$= \sigma(b_1 + w_{11}x_1 + w_{12}x_2 + w_{13}x_3)$

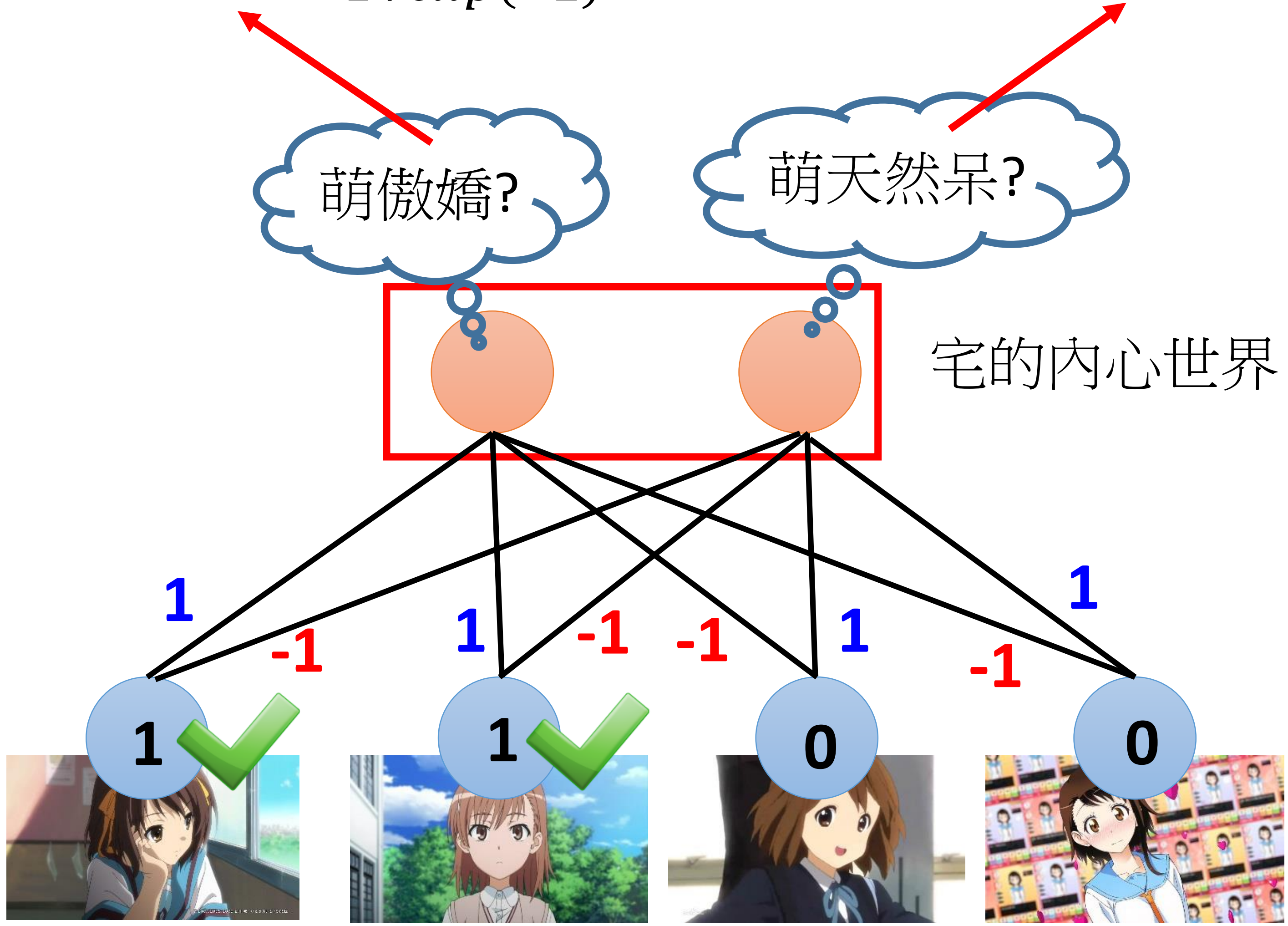
$P(h_2 = 1|x)$

$= \sigma(b_2 + w_{21}x_1 + w_{22}x_2 + w_{23}x_3)$

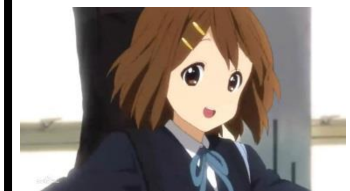
Given h_1, h_2 ……

Neural network with sigmoid function as activation function (誤)

$P(萌傲嬌=1|x) = \frac{1}{1+exp(-2)}$ $P(萌天然呆=1|x) = \frac{1}{1+exp(2)}$



RBM – Training without Labeling?

				
x^1	✓	✓		
x^2			✓	✓
x^3	✓	✓	✓	✓

Training data: $\{x^1, x^2 \cdots x^R\}$

Find the parameters which

maximize $\prod_{r=1}^{R} P(x^r)$

Maximizing the likelihood of observed data

RBM – Training without Labeling?

- Maximizing P(x)

$$P(x) = \sum_{h''} P(x, h'') = \sum_{h''} \frac{e^{E(x,h'')}}{\sum_{x',h'} e^{E(x',h')}} = \frac{\sum_{h''} e^{E(x,h'')}}{\sum_{x'} \sum_{h'} e^{E(x',h')}}$$

$$F(x) = \sum_{h''} e^{E(x,h'')} \quad \Rightarrow \quad P(x) = \frac{F(x)}{\sum_{x'} F(x')}$$

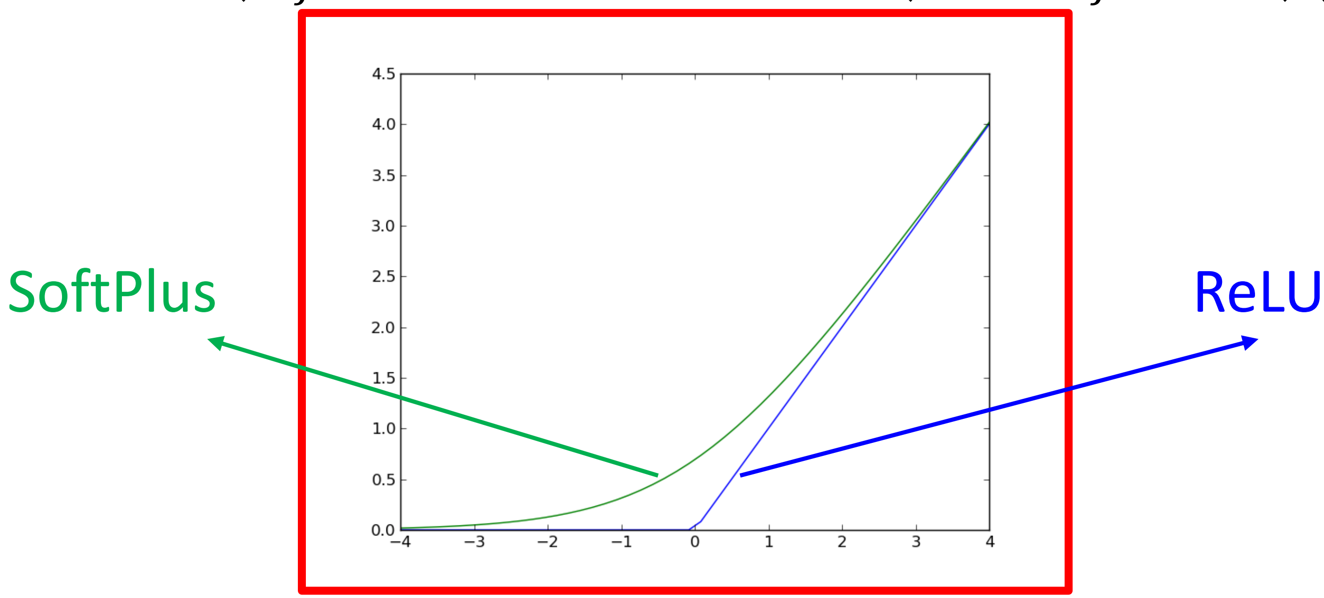
$$F(x) = exp\left(\sum_{x_j} c_j x_j + \sum_{h_i} log\left(1 + exp\left(b_i + \sum_{x_j} w_{ij} x_j\right)\right)\right)$$

(see the reference)

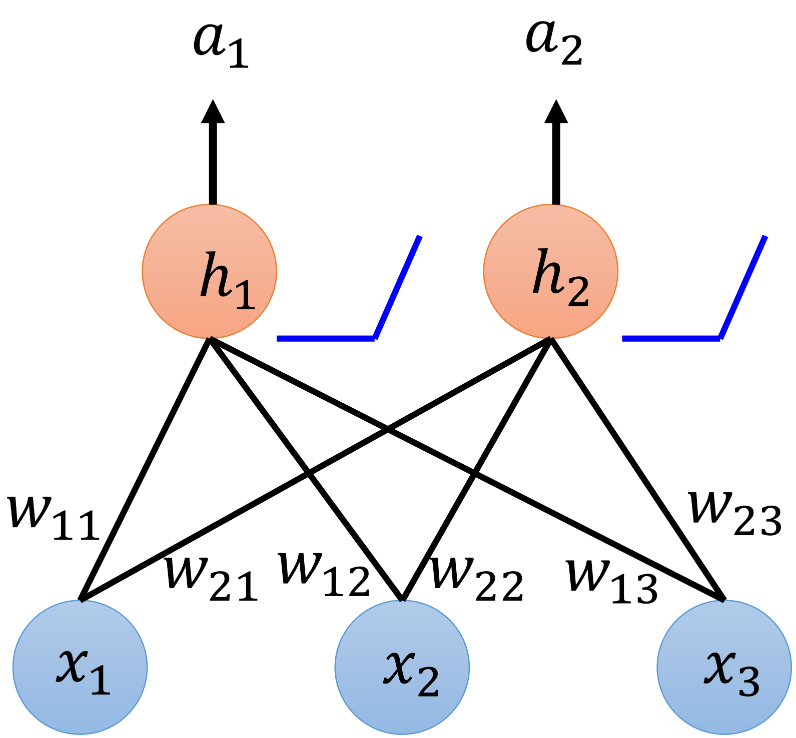
RBM – Training without Labeling?

$$F(x) = exp\left(\sum_{x_j} c_j x_j + \sum_{h_i} log\left(1 + exp\left(b_i + \sum_{x_j} w_{ij} x_j\right)\right)\right)$$

$$= exp\left(\sum_{x_j} c_j x_j + \sum_{h_i} softplus\left(b_i + \sum_{x_j} w_{ij} x_j\right)\right)$$



RBM – Training without Labeling?



Neural network with Softplus as activation function (誤)

$$F(x) = exp(a_1 + a_2)$$

The degree that the hidden layers are activated.

$$P(x) = \frac{F(x)}{\sum_{x'} F(x')}$$

$F(x)$ → Increase for x in data

$\sum_{x'} F(x')$ → Decrease for any x

RBM – Training by Gradient Ascent

- Given data $\{x^1, x^2 \cdots x^R\}$

maximize $\prod_{r=1}^{R} P(x^r)$ ⇒ maximize $\sum_{r=1}^{R} logP(x^r)$

$$P(x) = \sum_{h''} P(x, h'') = \sum_{h''} \frac{e^{E(x,h'')}}{\sum_{x',h'} e^{E(x',h')}} = \frac{\sum_{h''} e^{E(x,h'')}}{\sum_{x',h'} e^{E(x',h')}}$$

$$logP(x) = log \sum_{h''} e^{E(x,h'')} - log \sum_{x',h'} e^{E(x',h')}$$

$$w_{ij} \leftarrow w_{ij} + \eta \frac{\partial logP(x)}{\partial w_{ij}}$$

compute $\frac{\partial logP(x)}{\partial w_{ij}} = ?$

Warning of Math

$$\log P(x) = \underbrace{\log \sum_{h''} e^{E(x,h'')}}_{A} - \underbrace{\log \sum_{x',h'} e^{E(x',h')}}_{B} \qquad \frac{\partial \log P(x)}{\partial w_{ij}} = ?$$

$$E(x,h) = \sum_{h_i} b_i h_i + \sum_{x_j} c_j x_j + \sum_{h_i,x_j} w_{ij} h_i x_j \qquad \frac{\partial E(x,h)}{\partial w_{ij}} = h_i x_j$$

$$\frac{\partial A}{\partial w_{ij}} = \frac{1}{\sum_{h''} e^{E(x,h'')}} \sum_{h''} \frac{\partial e^{E(x,h'')}}{\partial w_{ij}}$$

$$= \frac{1}{\sum_{h''} e^{E(x,h'')}} \sum_{h''} e^{E(x,h'')} \frac{\partial E(x,h'')}{\partial w_{ij}}$$

$$= \frac{1}{\sum_{h''} e^{E(x,h'')}} \sum_{h''} e^{E(x,h'')} h_i'' x_j = \sum_{h''} \underbrace{\frac{e^{E(x,h'')}}{\sum_{h''} e^{E(x,h'')}}}_{P(h''|x)} h_i'' x_j$$

$$= \sum_{h''} P(h''|x) h_i'' x_j$$

$$\log P(x) = \underbrace{\log \sum_{h''} e^{E(x,h'')}}_{A} - \underbrace{\log \sum_{x',h'} e^{E(x',h')}}_{B} \qquad \frac{\partial \log P(x)}{\partial w_{ij}} = ?$$

$$\frac{\partial B}{\partial w_{ij}} = \frac{1}{\sum_{x',h'} e^{E(x',h')}} \sum_{x',h'} \frac{\partial e^{E(x',h')}}{\partial w_{ij}}$$

$$= \frac{1}{\sum_{x',h'} e^{E(x',h')}} \sum_{x',h'} e^{E(x',h')} \frac{\partial E(x',h')}{\partial w_{ij}}$$

$$= \frac{1}{\sum_{x',h'} e^{E(x',h')}} \sum_{x',h'} e^{E(x',h')} h'_i x'_j$$

$$= \sum_{x',h'} \frac{e^{E(x',h')}}{\sum_{x',h'} e^{E(x',h')}} h'_i x'_j = \sum_{x',h'} P(x',h') h'_i x'_j$$

End of Warning

$$logP(x) = \underbrace{log \sum_{h''} e^{E(x,h'')}}_{A} - \underbrace{log \sum_{x',h'} e^{E(x',h')}}_{B} \qquad \frac{\partial logP(x)}{\partial w_{ij}} = ?$$

$$\frac{\partial A}{\partial w_{ij}} = \sum_{h''} P(h''|x) h_i'' x_j = E_{P(h''|x)}[h_i'' x_j]$$

The expected value of $h_i'' x_j$ based on $P(h''|x)$ given the current parameters

$$logP(x) = \underbrace{log \sum_{h''} e^{E(x,h'')}}_{A} - \underbrace{log \sum_{x',h'} e^{E(x',h')}}_{B} \qquad \frac{\partial logP(x)}{\partial w_{ij}} = ?$$

$$\frac{\partial A}{\partial w_{ij}} = \sum_{h''} P(h''|x) h_i'' x_j = E_{P(h''|x)}[h_i'' x_j]$$

The expected value of $h_i'' x_j$ based on $P(h''|x)$ given the current parameters

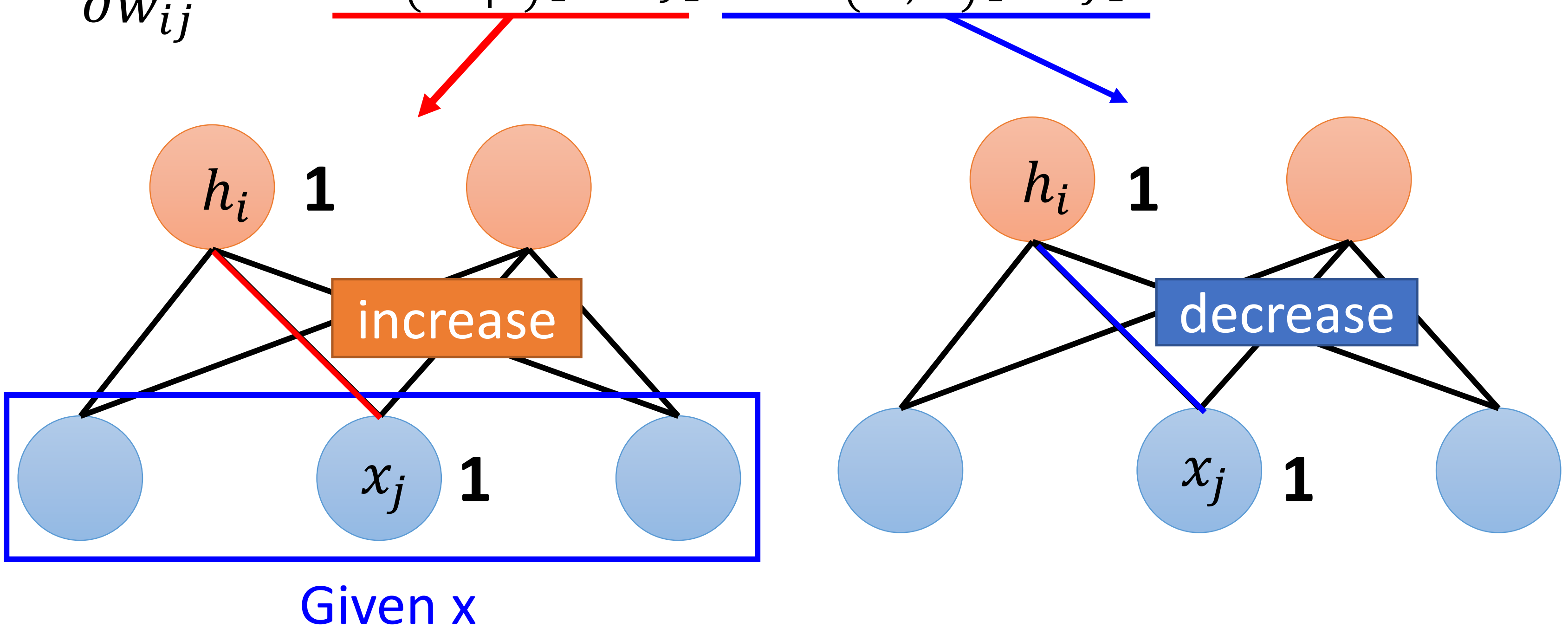
$$\frac{\partial B}{\partial w_{ij}} = \sum_{x',h'} P(x',h') h_i' x_j' = E_{P(x',h')}[h_i' x_j']$$

The expected value of $h_i' x_j'$ based on $P(x',h')$ given the current parameters

RBM
– Training by Gradient Ascent

$$w_{ij} \leftarrow w_{ij} + \eta \frac{\partial logP(x)}{\partial w_{ij}}$$

$$\frac{\partial logP(x)}{\partial w_{ij}} = E_{P(h''|x)}[h_i'' x_j] - E_{P(x',h')}[h_i' x_j']$$



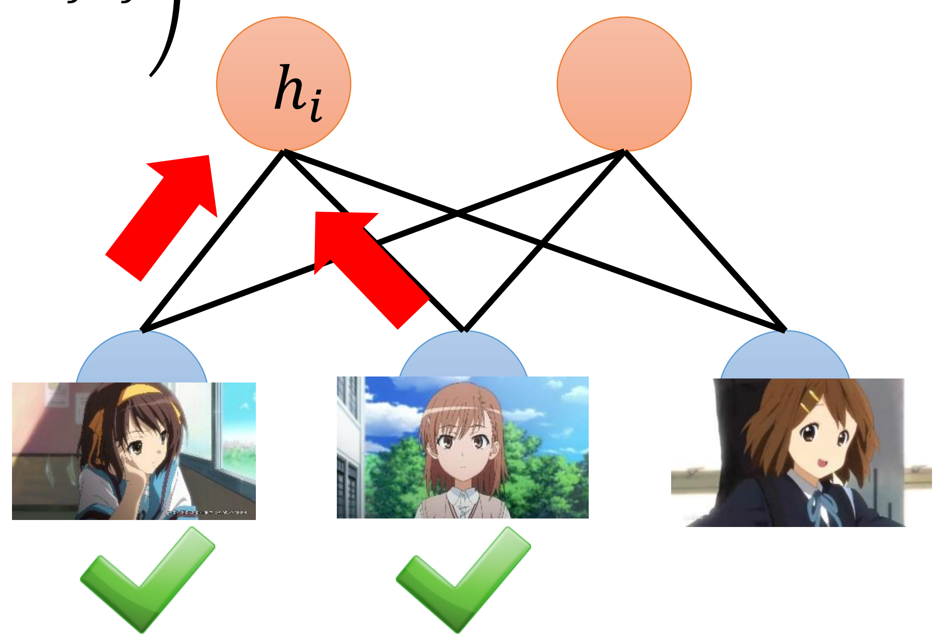
$$\log P(x) = \underbrace{\log \sum_{h''} e^{E(x,h'')}}_{A} - \underbrace{\log \sum_{x',h'} e^{E(x',h')}}_{B}$$

$$\frac{\partial \log P(x)}{\partial w_{ij}} = ?$$

$$\frac{\partial A}{\partial w_{ij}} = \sum_{h''} P(h''|x) h_i'' x_j = E_{P(h''|x)}[h_i'' x_j]$$

$$= P(h_i'' = 1|x) x_j$$ Just a sigmoid

$$P(h_i'' = 1|x) = sig\left(b_i + \sum_{x_j} w_{ij} x_j\right)$$



$$logP(x) = \underbrace{log\sum_{h''} e^{E(x,h'')}}_{A} - \underbrace{log\sum_{x',h'} e^{E(x',h')}}_{B} \qquad \frac{\partial logP(x)}{\partial w_{ij}} = ?$$

$$\frac{\partial A}{\partial w_{ij}} = \sum_{h''} P(h''|x) h_i'' x_j = E_{P(h''|x)}[h_i'' x_j]$$

$$= P(h_i'' = 1|x) x_j$$ Just a sigmoid

$$\frac{\partial B}{\partial w_{ij}} = \sum_{x',h'} P(x',h') h_i' x_j' = E_{P(x',h')}[h_i' x_j']$$

Exact computing is not tractable

Sample by Gibbs sampling

$$x^1, h^1 \sim P(x',h')$$

$$x^2, h^2 \sim P(x',h')$$

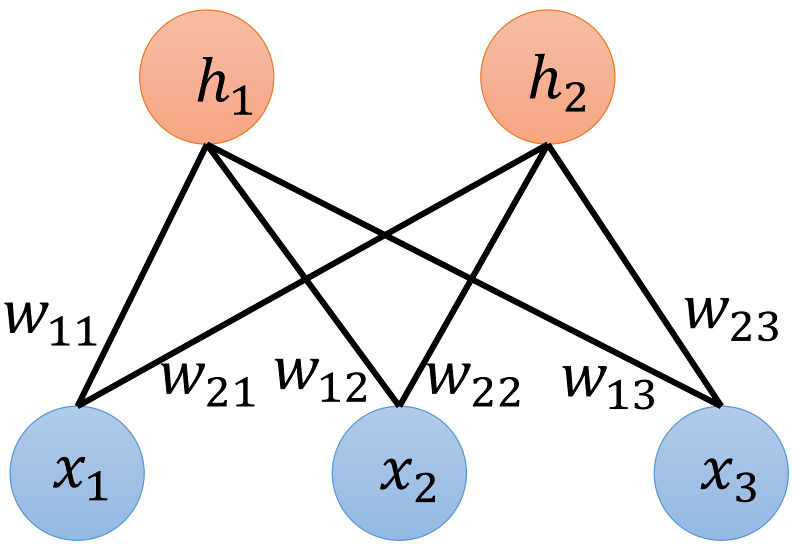
$$\vdots$$

$$x^N, h^N \sim P(x',h')$$

$$\frac{1}{N}\sum_{n=1}^{N} h_i^n x_j^n$$

RBM –Gibbs Sampling

Use Gibbs sampling to sample from P(x,h)



Random initialize x^0, h^0

For n = 1 to N

$x_1^n \sim P(x_1 | \cancel{x_2^{n-1}, x_3^{n-1}}, h_1^{n-1}, h_2^{n-1})$

$x_2^n \sim P(x_2 | \cancel{x_1^n, x_3^{n-1}}, h_1^{n-1}, h_2^{n-1})$

$x_3^n \sim P(x_3 | \cancel{x_1^n, x_2^n}, h_1^{n-1}, h_2^{n-1})$

$h_1^n \sim P(h_1 | x_1^n, x_2^n, x_3^n, \cancel{h_2^{n-1}})$

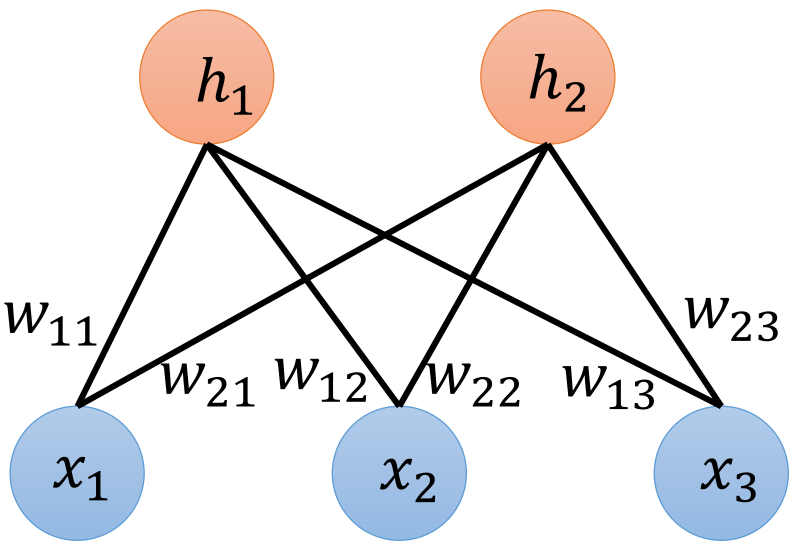
$h_2^n \sim P(h_2 | x_1^n, x_2^n, x_3^n, \cancel{h_1^n})$

Obtain one sample x^n, h^n

(as sample from P(x,h))

RBM –Gibbs Sampling

Use Gibbs sampling
to sample from P(x,h)



Random initialize x^0, h^0

For n = 1 to N

$x_1^n \sim P(x_1 | h_1^{n-1}, h_2^{n-1})$

$x_2^n \sim P(x_2 | h_1^{n-1}, h_2^{n-1})$

$x_3^n \sim P(x_3 | h_1^{n-1}, h_2^{n-1})$ $\quad x^n \sim P(x | h^{n-1})$

$h_1^n \sim P(h_1 | x_1^n, x_2^n, x_3^n)$

$h_2^n \sim P(h_2 | x_1^n, x_2^n, x_3^n)$ $\quad h^n \sim P(h | x^n)$

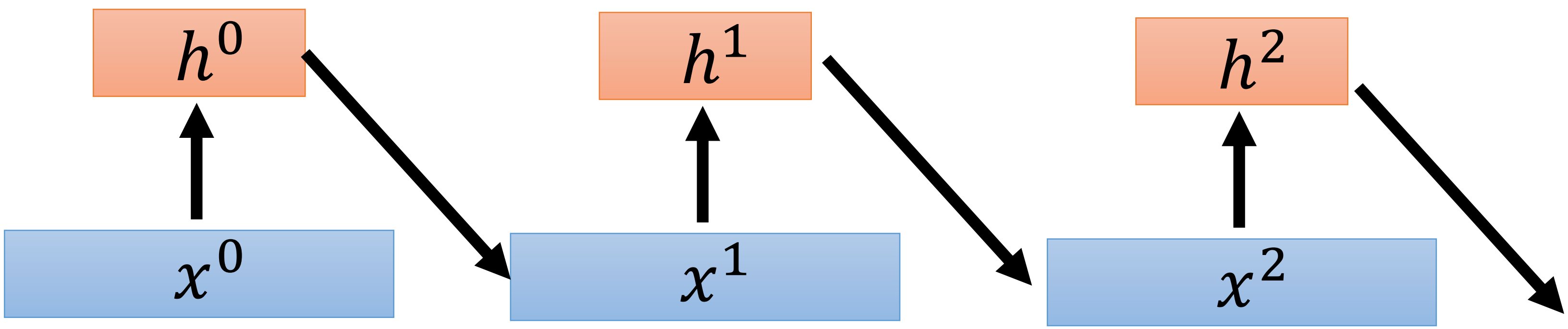
Obtain one sample x^n, h^n

(as sample from P(x,h))

RBM –Gibbs Sampling

$x^n \sim P(x|h^{n-1})$

$h^n \sim P(h|x^n)$

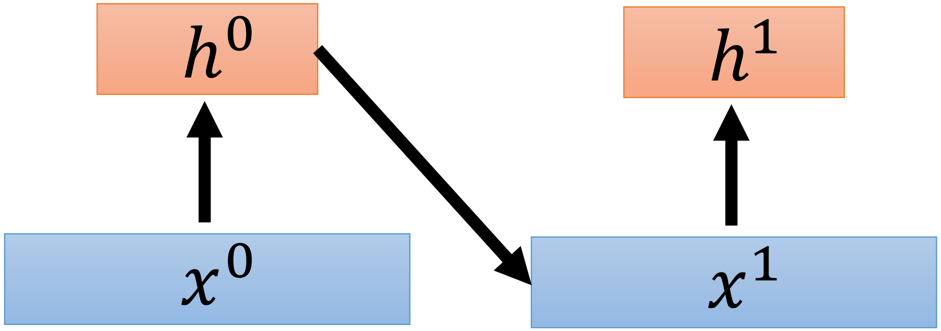


x^0 is data x

$$E_{P(x',h')}[h'_i x'_j] \approx \frac{1}{N}\sum_{n=1}^{N} h_i^n x_j^n$$

Each time we update parameters, we should do Gibbs sampling?!

RBM – Contrastive Divergence (CD)



Stop!

x^0 is data x

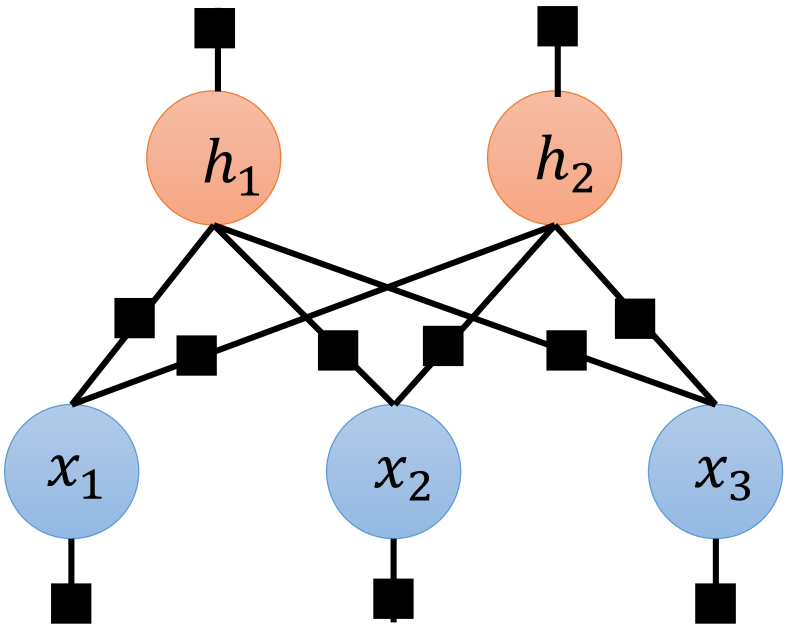
$$E_{P(x',h')}\left[h_i' x_j'\right] \approx h_i^1 x_j^1$$

It works in reality!

Persistent CD

Ref: Tieleman, Tijmen. "Training restricted Boltzmann machines using approximations to the likelihood gradient." *Proceedings of the 25th international conference on Machine learning*. ACM, 2008.

RBM - Generalization



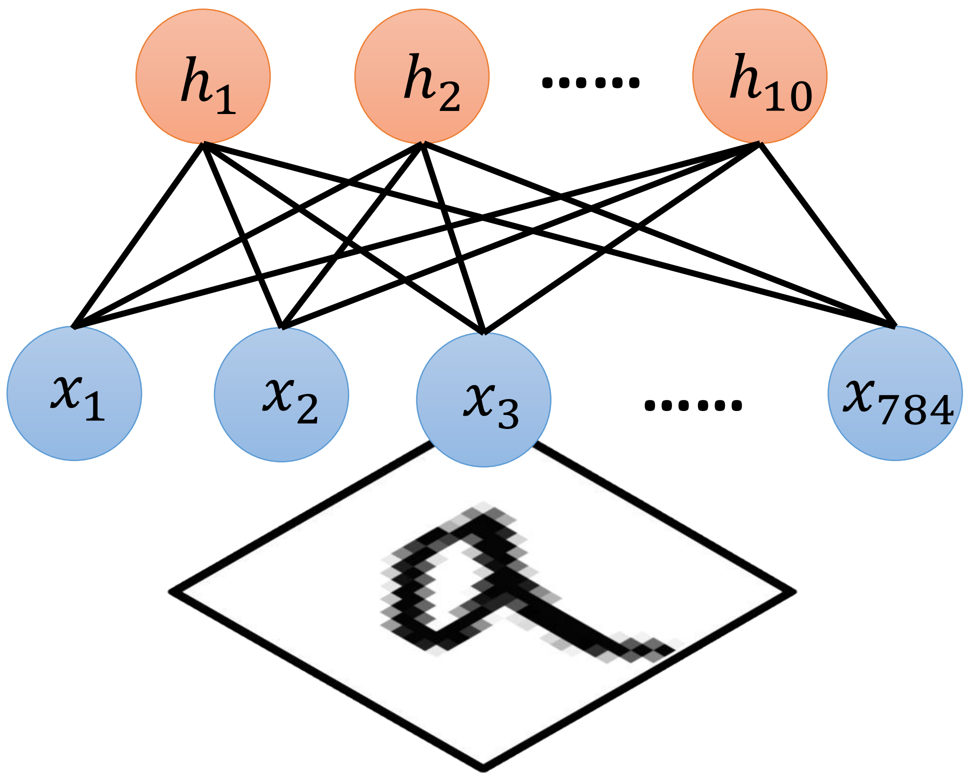
~~$g_j(x_j) = \begin{cases} c_j & x_j = 1 \\ 0 & else \end{cases}$~~

If x are real numbers

$$g_j(x_j) = -\left(x_j - c_j\right)^2$$

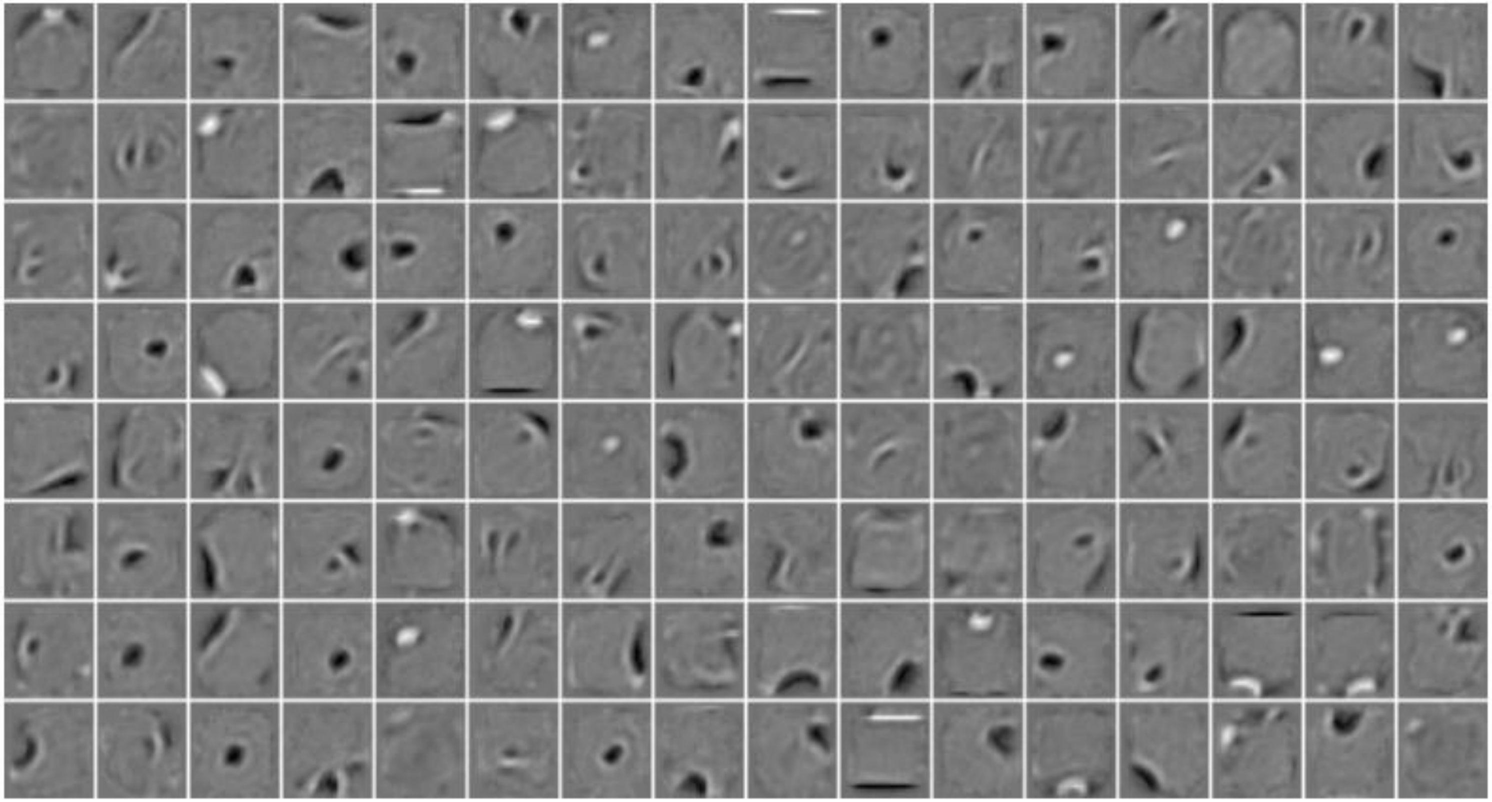
$$E(x,h) = \sum_{h_i} b_i h_i + \sum_{h_i,x_j} w_{ij} h_i x_j - \sum_{x_j} \left(x_j - c_j\right)^2$$

RBM - Handwritten Digits



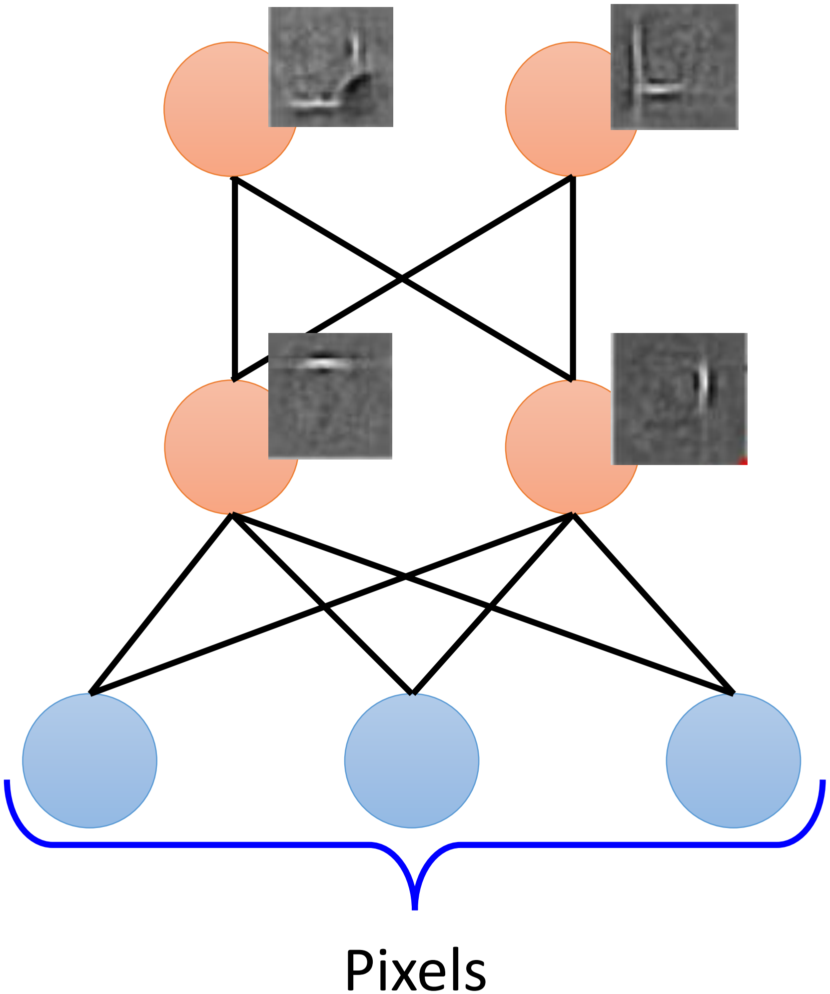
Pixels in handwriting image

RBM – Handwritten Digits

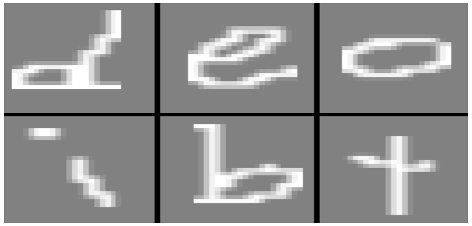


Source of image: Larochelle, H., Bengio, Y., Louradour, J., & Lamblin, P. (2009). Exploring strategies for training deep neural networks. *The Journal of Machine Learning Research*, *10*, 1-40.

Deep Boltzmann Machines

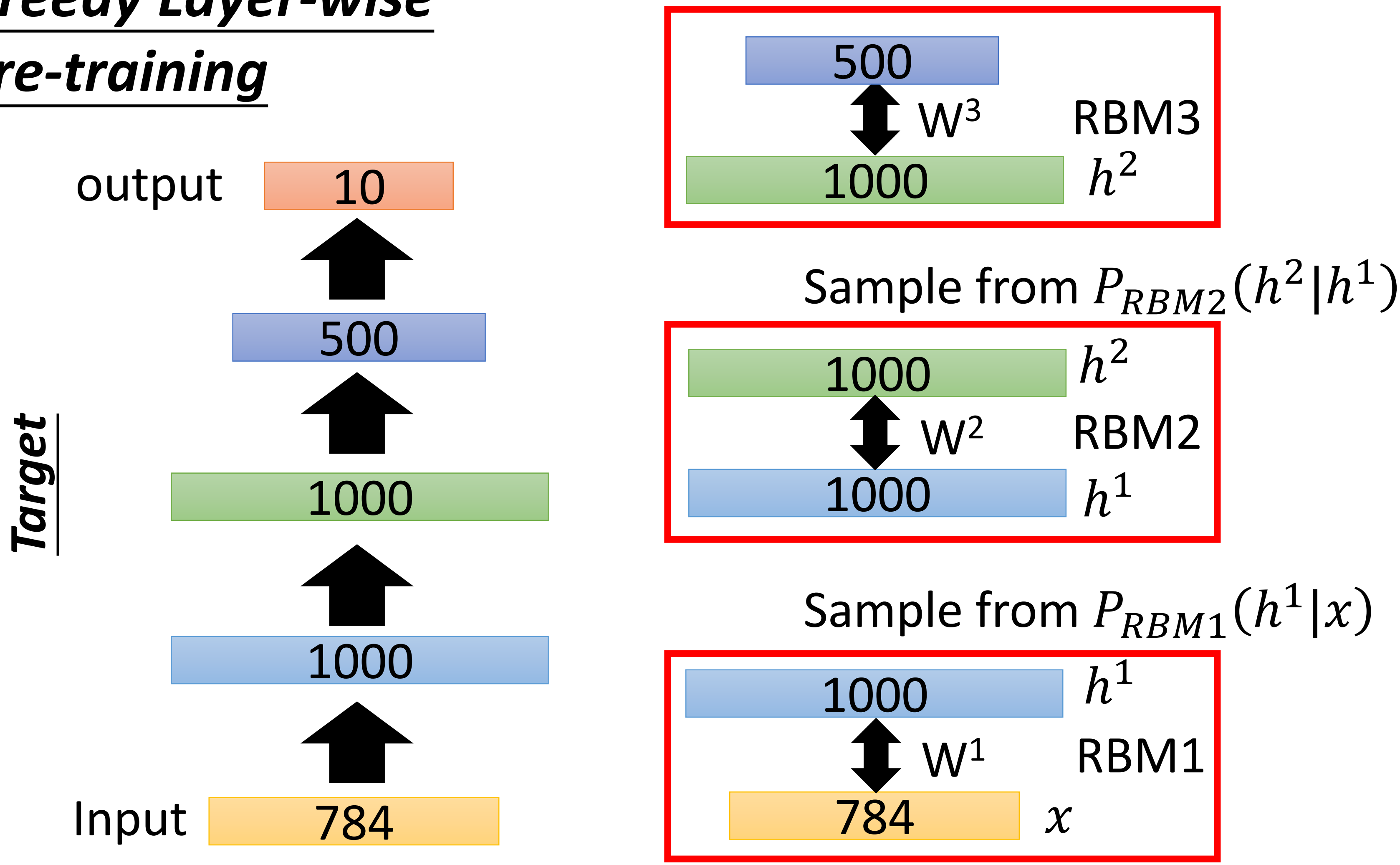


Ref: Salakhutdinov, Ruslan, and Geoffrey E. Hinton. "Deep Boltzmann machines.“ *International Conference on Artificial Intelligence and Statistics*. 2009.



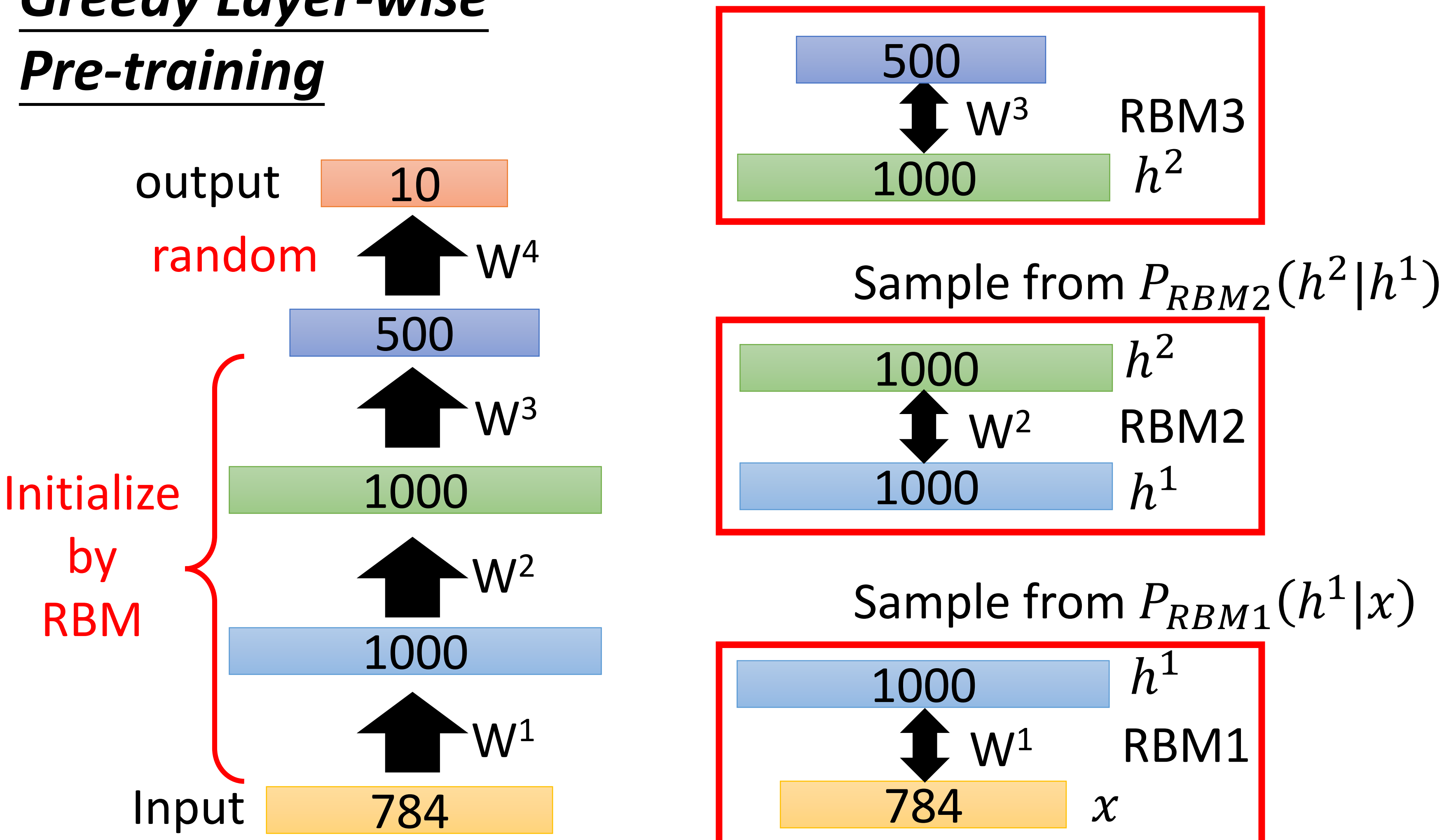
RBM - Pre-training DNN

Greedy Layer-wise Pre-training



RBM - Pre-training DNN

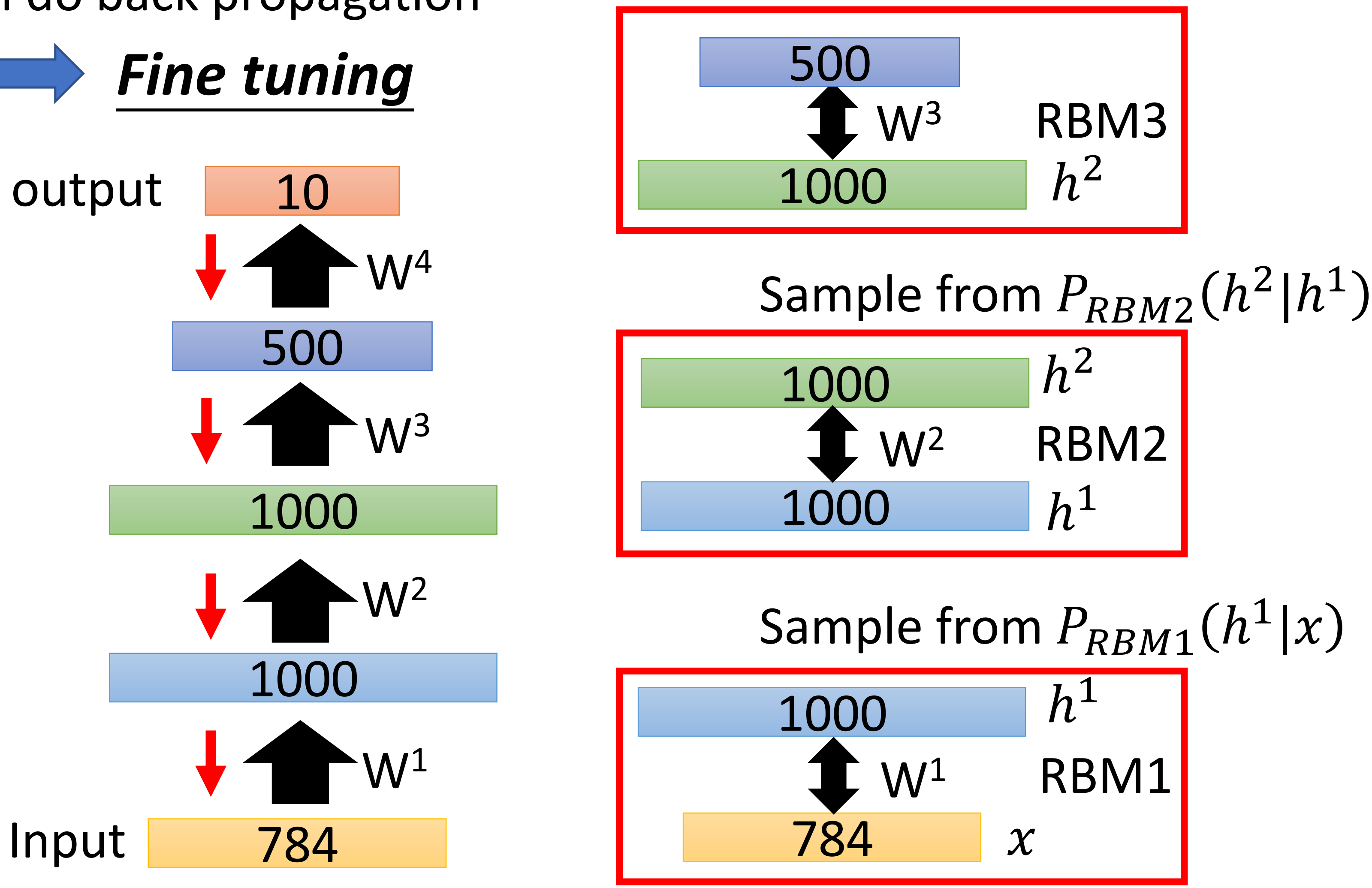
Greedy Layer-wise Pre-training



RBM - Pre-training DNN

Then do back propagation

→ ***Fine tuning***

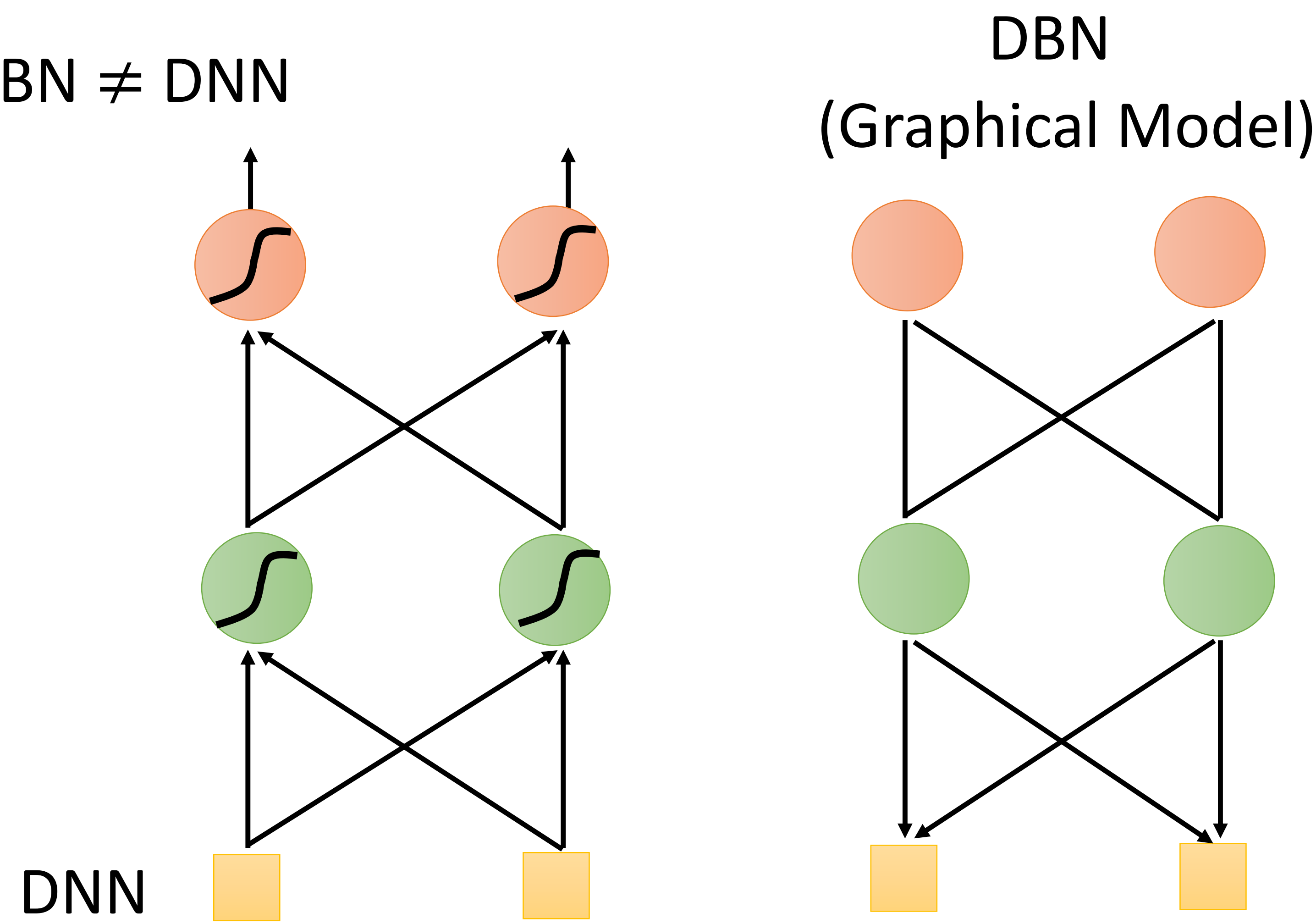


Reference for RBM

- Independent
  - Neural networks [5.2] : Restricted Boltzmann machine – inference
  - https://www.youtube.com/watch?v=lekCh_i32iE&list=PL6Xpj9I5qXYEcOhn7TqghAJ6NAPrNmUBH&index=37
- Intuition for maximizing likelihood
  - Neural networks [5.3] : Restricted Boltzmann machine - free energy
  - https://www.youtube.com/watch?v=e0Ts_7Y6hZU&list=PL6Xpj9I5qXYEcOhn7TqghAJ6NAPrNmUBH&index=38

Deep Belief Network (DBN)

- DBN ≠ DNN



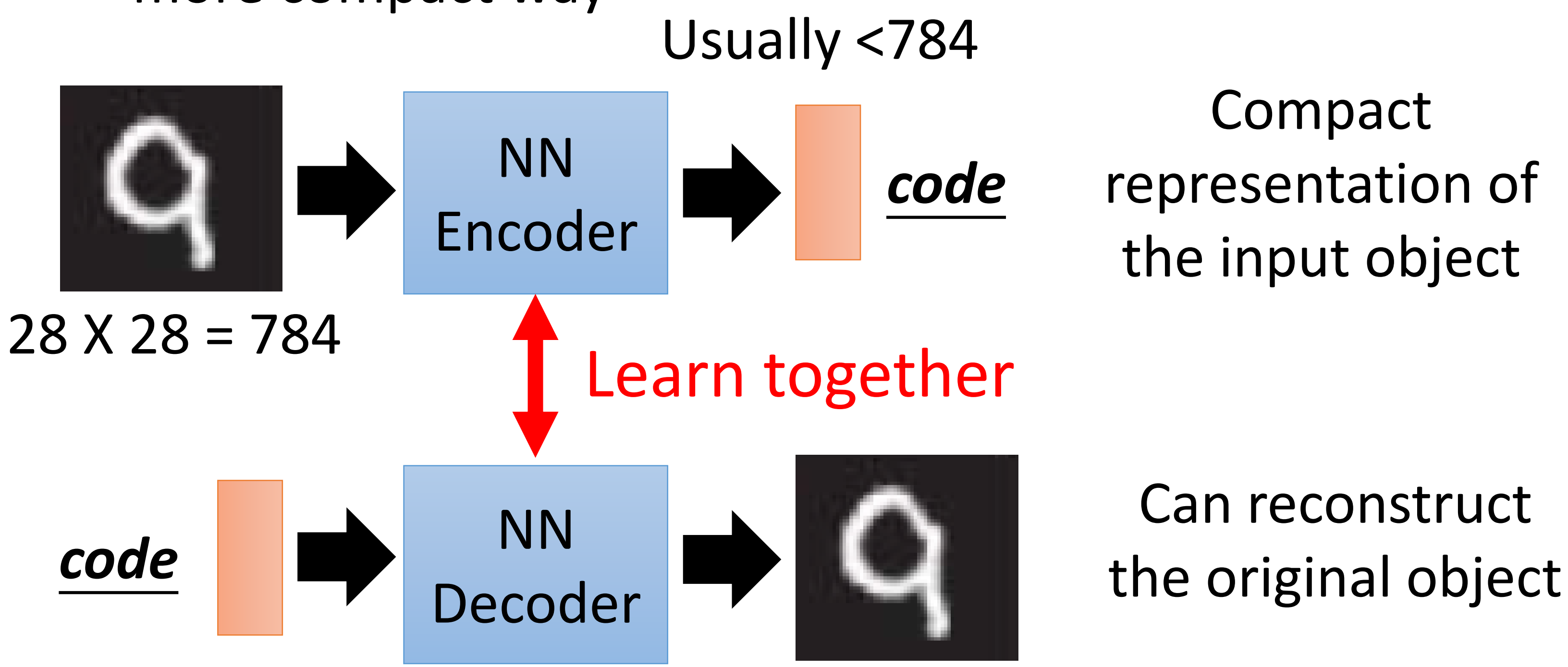
Reference for DBN

- Neural networks [7.7] : Deep learning - deep belief network
  - https://www.youtube.com/watch?v=vkb6AWYXZ5I&list=PL6Xpj9I5qXYEcOhn7TqghAJ6NAPrNmUBH&index=57
- Neural networks [7.8] : Deep learning - variational bound
  - https://www.youtube.com/watch?v=pStDscJh2Wo&list=PL6Xpj9I5qXYEcOhn7TqghAJ6NAPrNmUBH&index=58
- Neural networks [7.9] : Deep learning - DBN pre-training
  - https://www.youtube.com/watch?v=35MUlYCColk&list=PL6Xpj9I5qXYEcOhn7TqghAJ6NAPrNmUBH&index=59

Auto-encoder

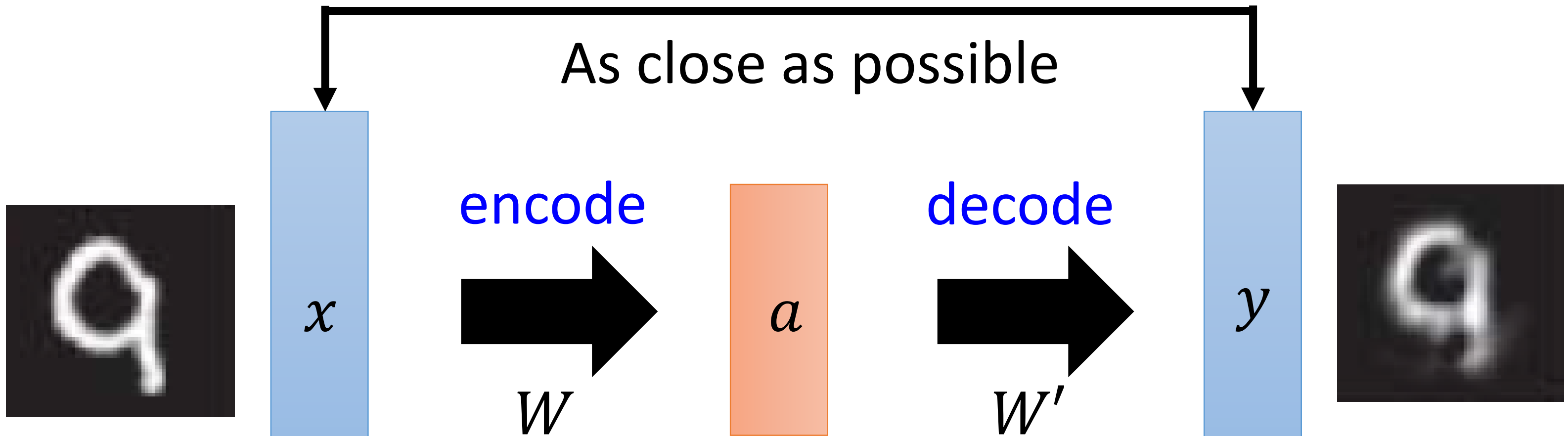
Auto-encoder

- We use 28 X 28 d to represent a digit
- Not all 28 X 28 images are digit
- It is possible to represent the images of digits in a more compact way



Auto-encoder

Minimize $(x - y)^2$



Input layer

hidden layer

Bottleneck later

output layer

$$a = \sigma(Wx + b) \qquad y = \sigma(W'a + b')$$

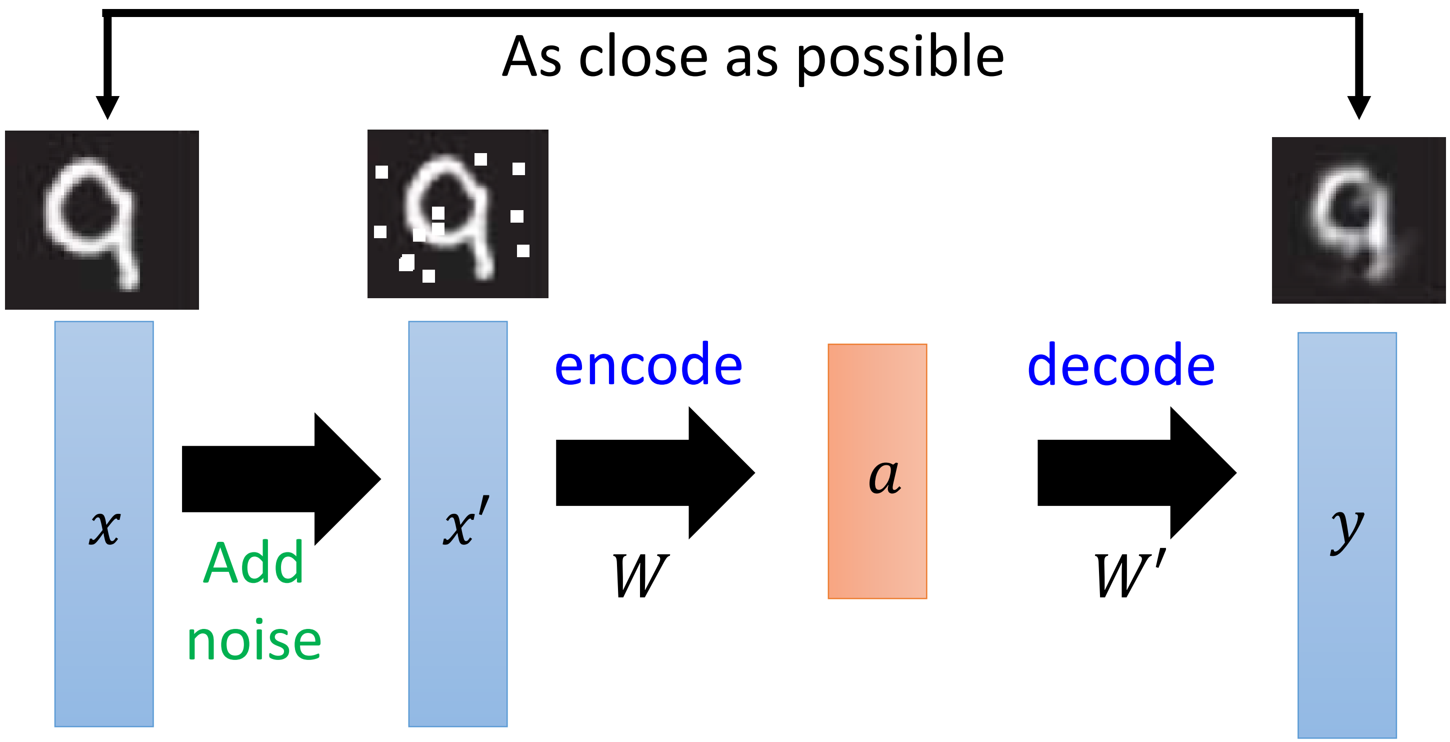
Output of the hidden layer is the code

More: Contractive auto-encoder

Ref: Rifai, Salah, et al. "Contractive auto-encoders: Explicit invariance during feature extraction.“ *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*. 2011.

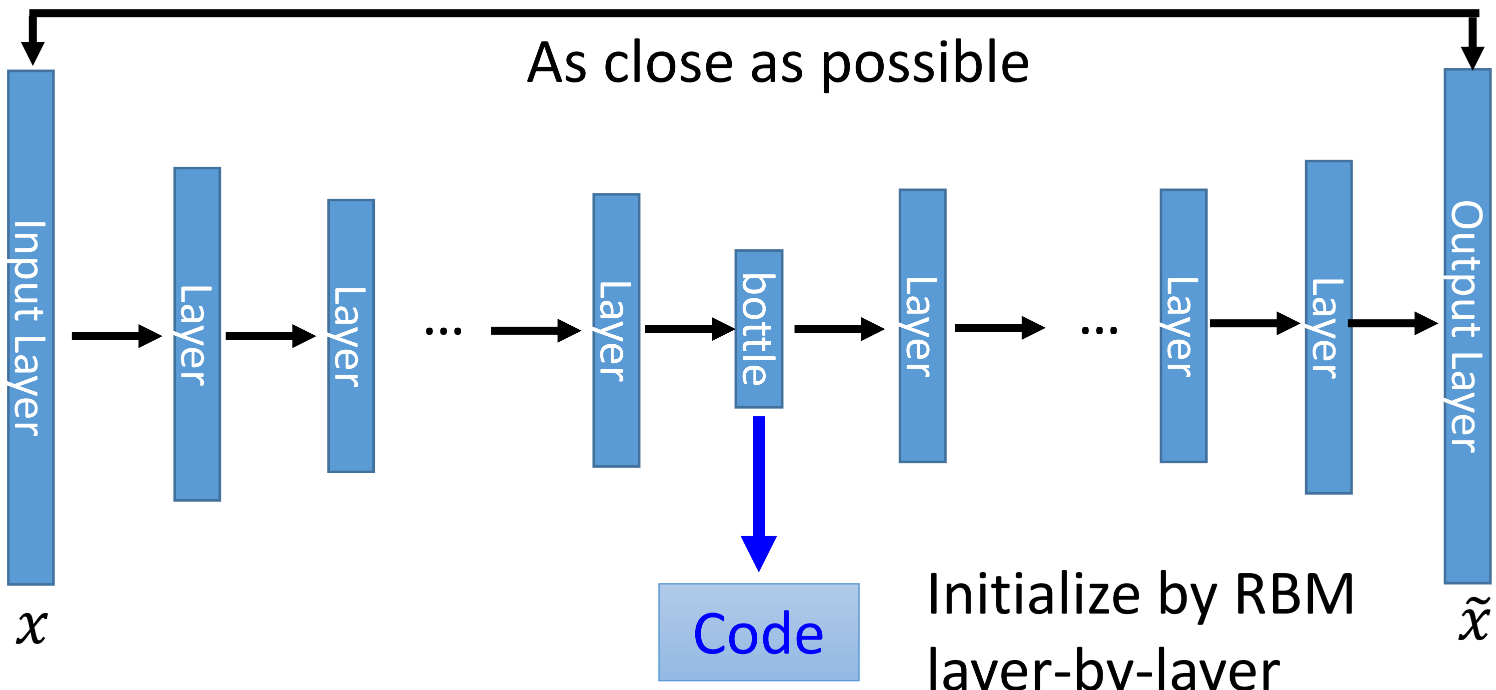
Auto-encoder

- De-noising auto-encoder



Deep Auto-encoder

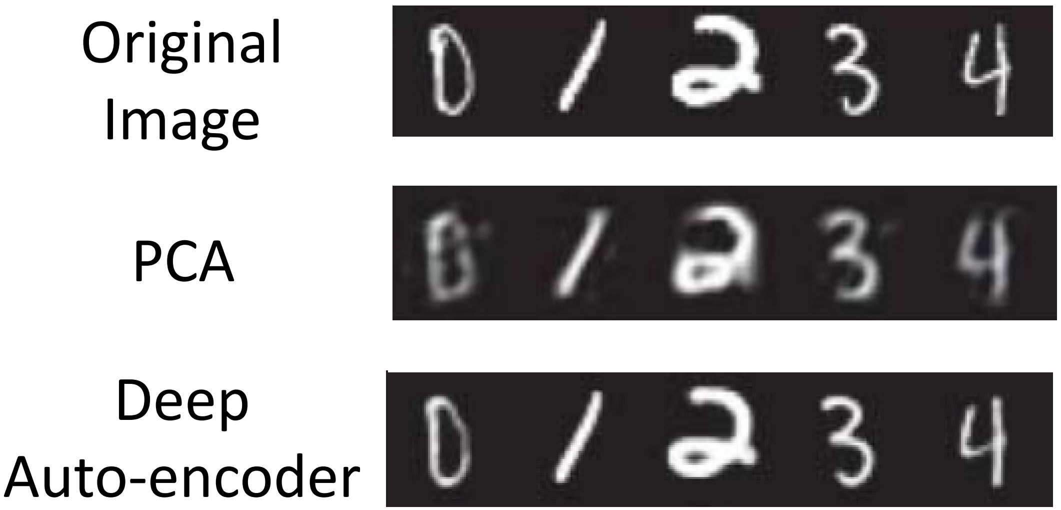
- Of course, the auto-encoder can be deep

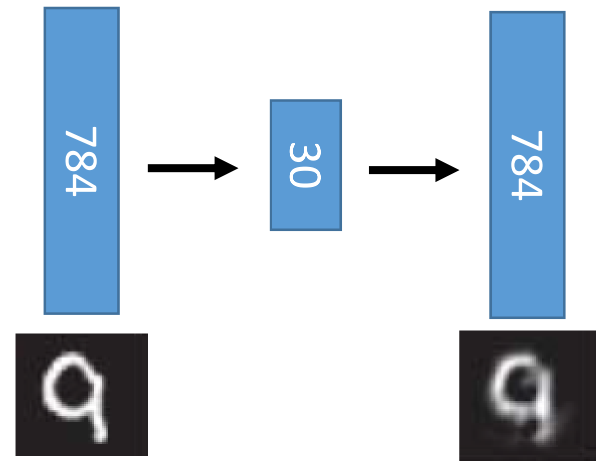


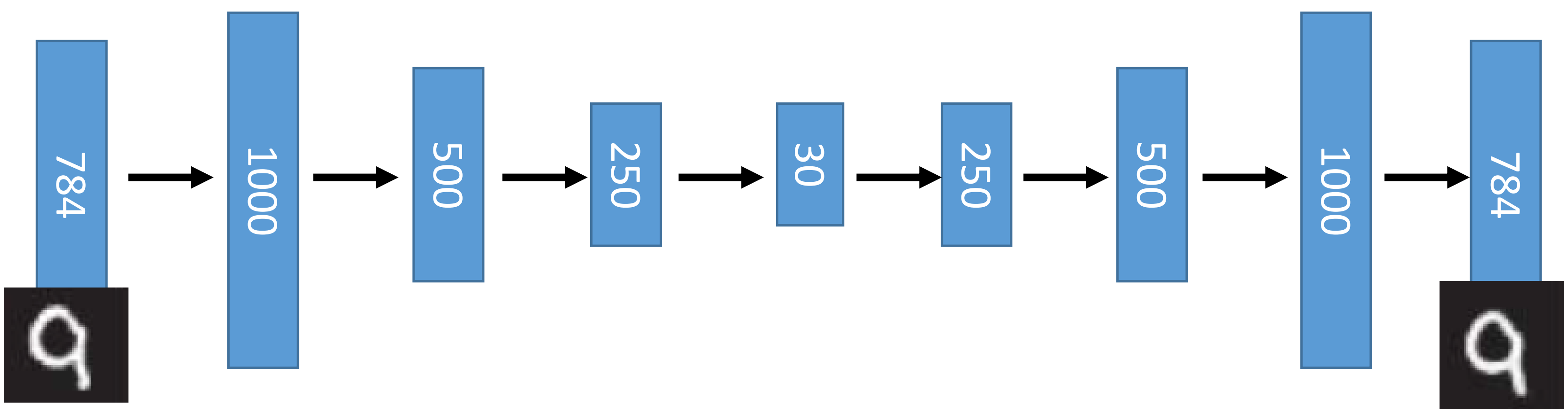
Initialize by RBM layer-by-layer

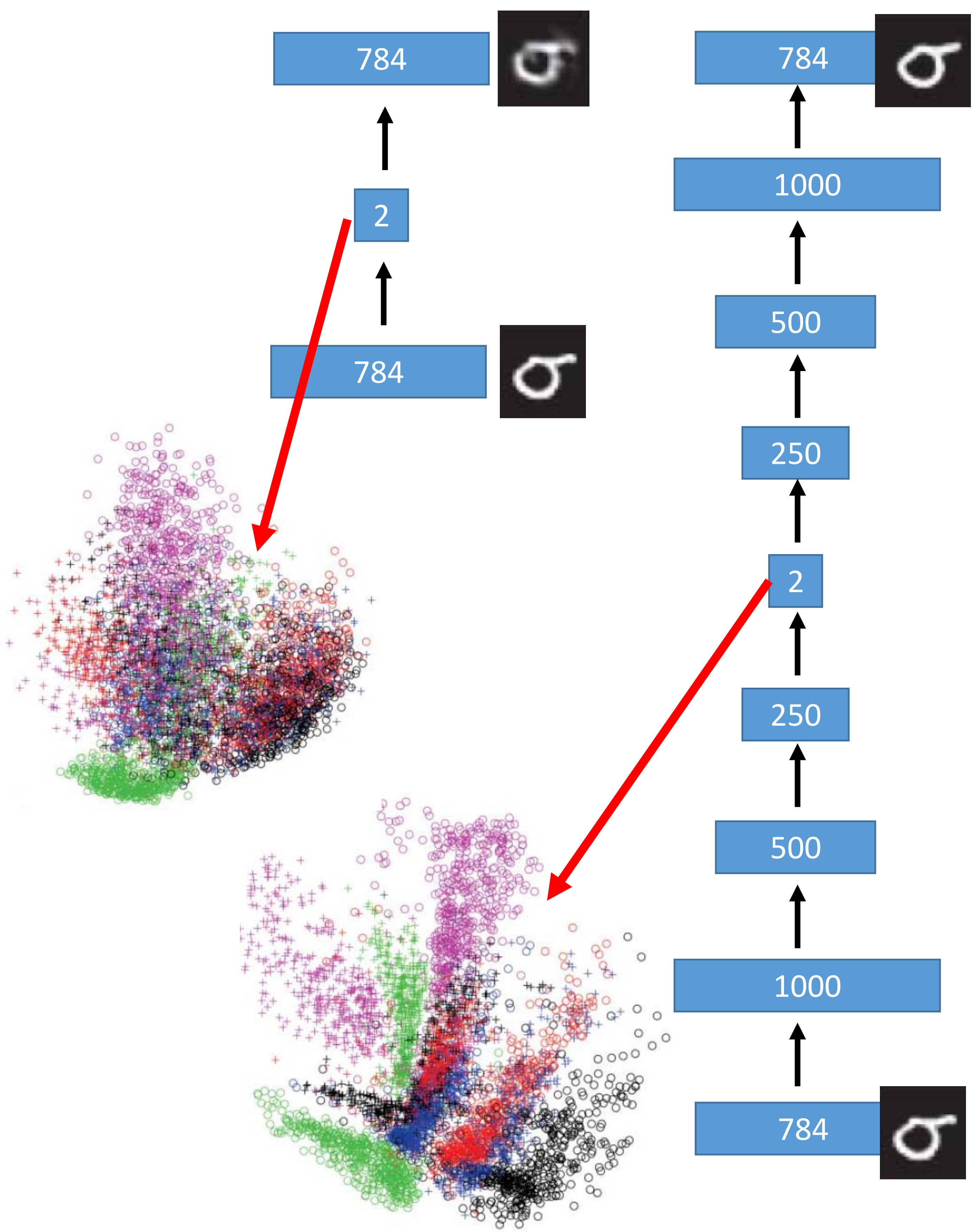
Reference: Hinton, Geoffrey E., and Ruslan R. Salakhutdinov. "Reducing the dimensionality of data with neural networks." *Science* 313.5786 (2006): 504-507

Deep Auto-encoder



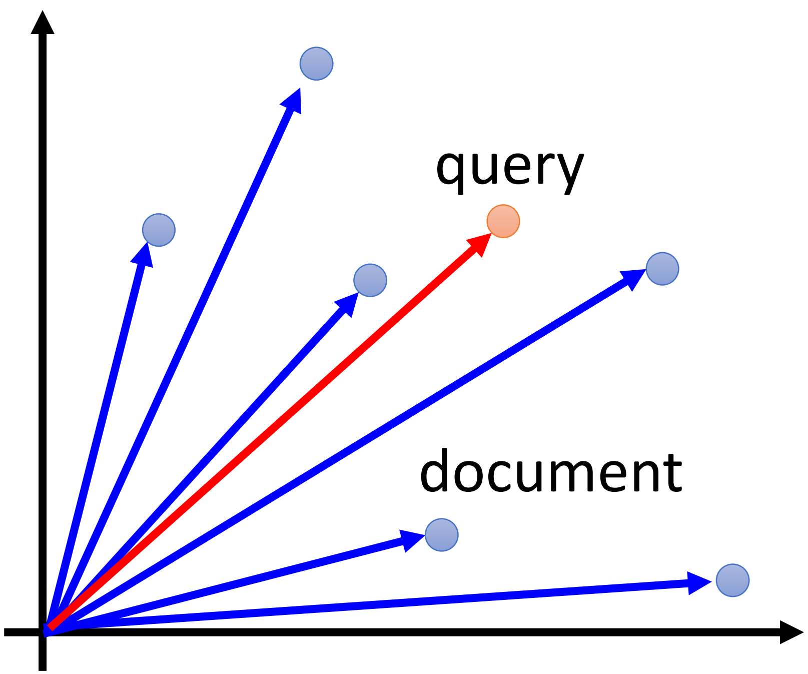





784
2
784
784
1000
500
250
2
250
500
1000
784

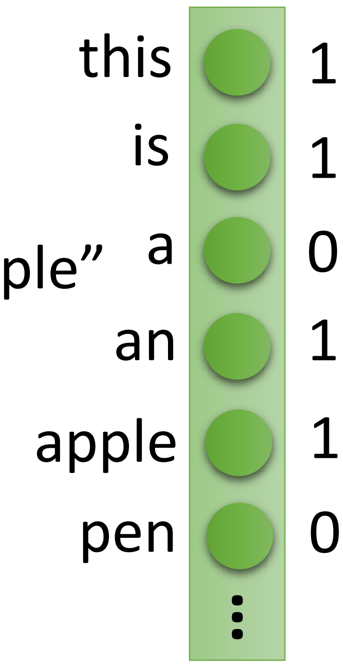
Auto-encoder – Text Retrieval

Vector Space Model



Bag-of-word

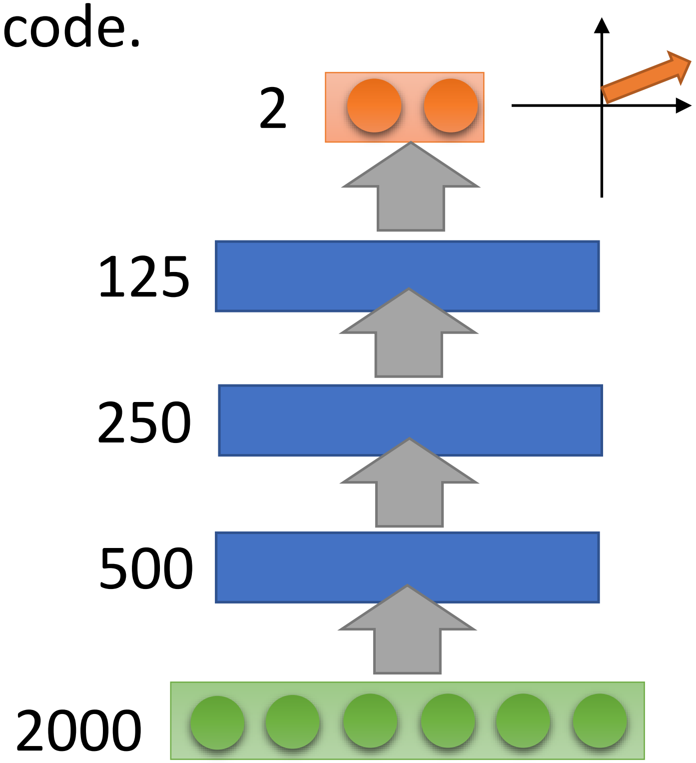
word string:
"This is an apple"



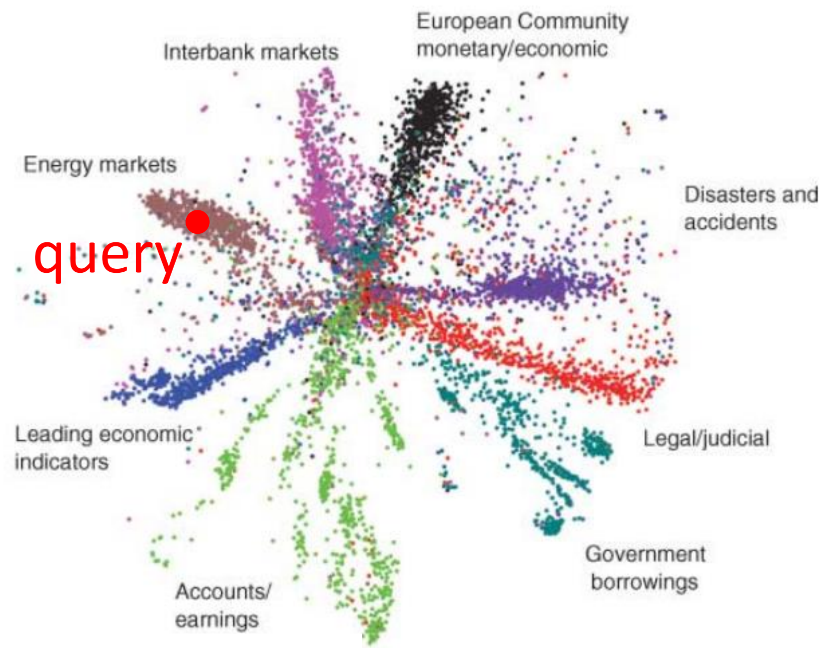
Semantics are not considered.

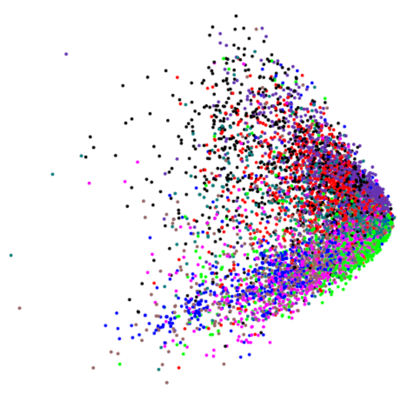
Auto-encoder – Text Retrieval

The documents talking about the same thing will have close code.



Bag-of-word
(document or query)





LSA: project documents to 2 latent topics

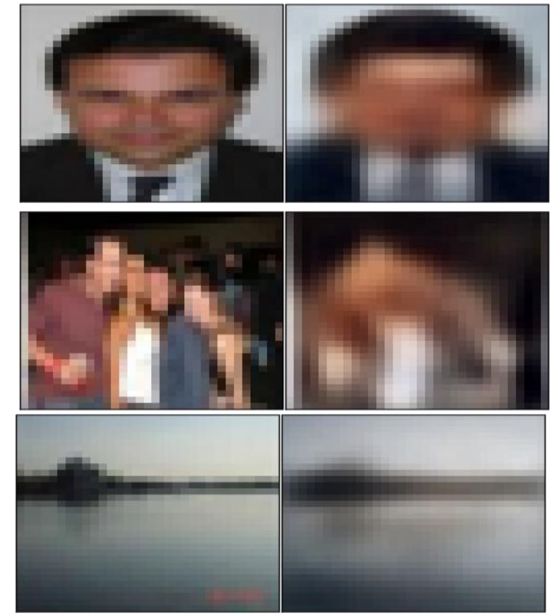
Auto-encoder – Similar Image Search

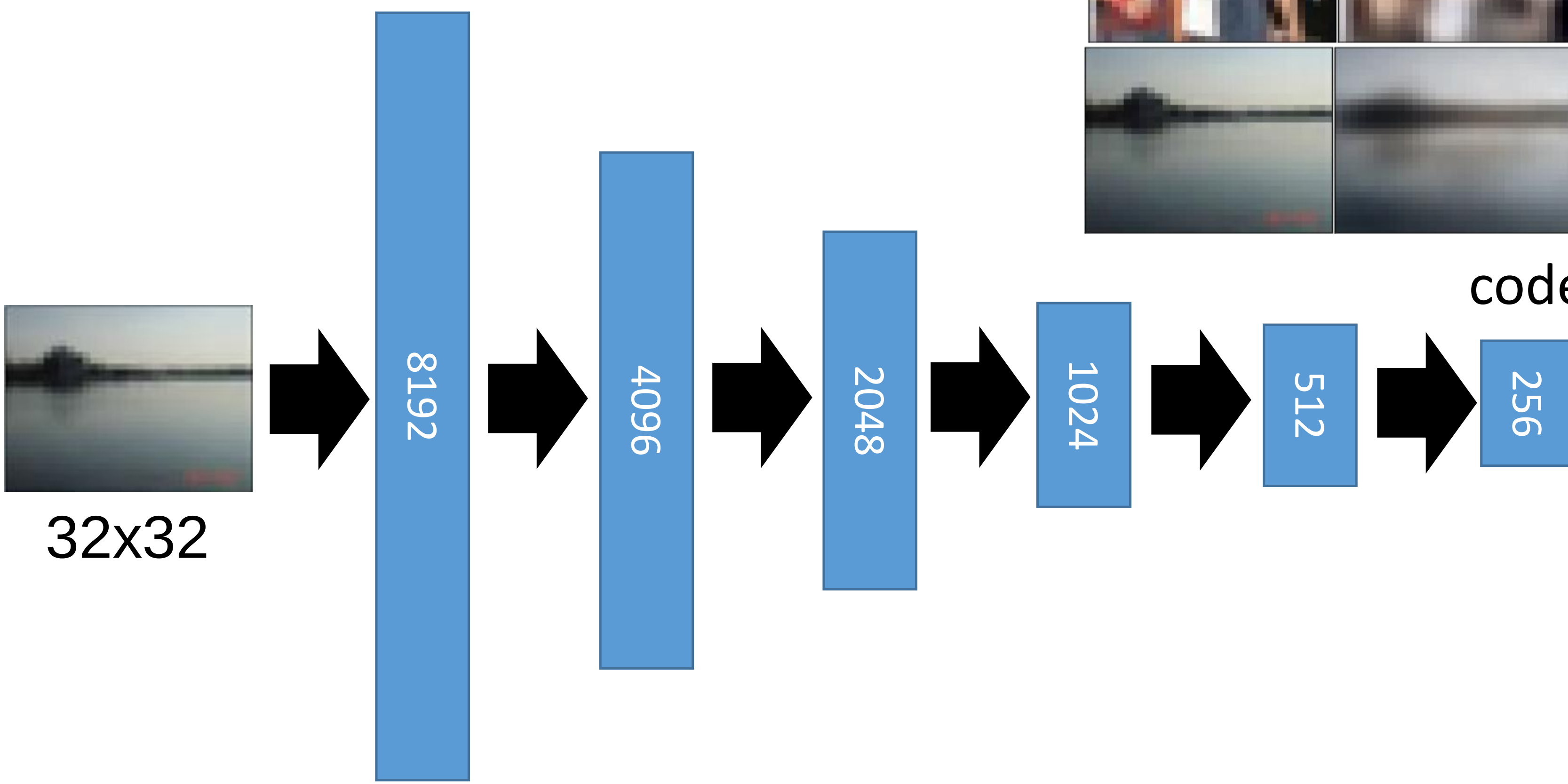
Retrieved using Euclidean distance in pixel intensity space

(Images from Hinton's slides on Coursera)

Reference: Krizhevsky, Alex, and Geoffrey E. Hinton. "Using very deep autoencoders for content-based image retrieval." *ESANN*. 2011.

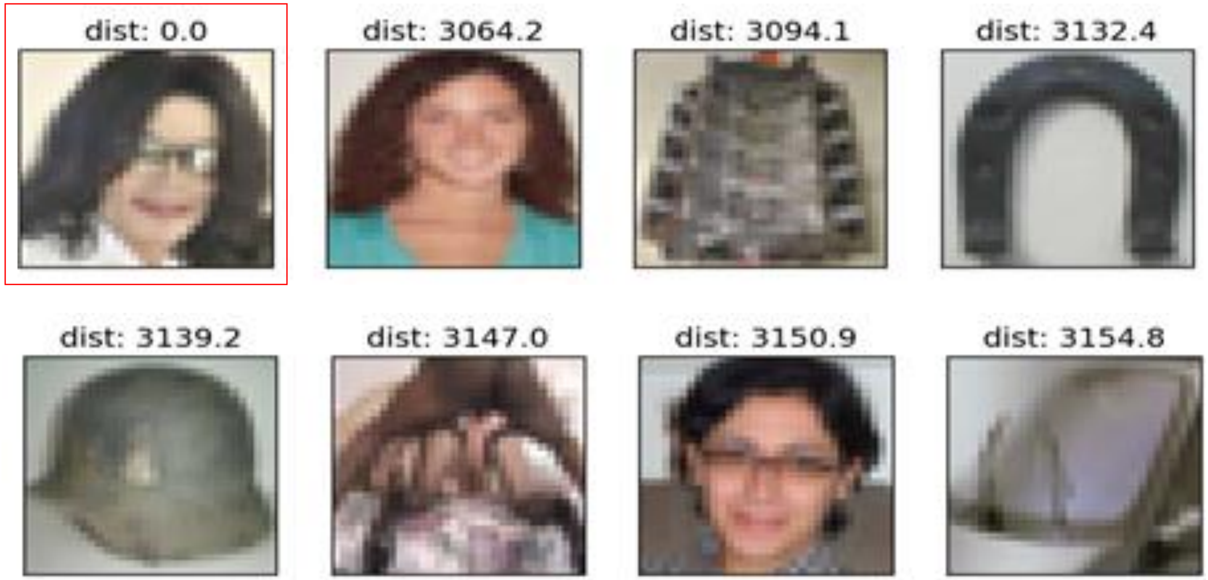
Auto-encoder – Similar Image Search





(crawl millions of images from the Internet)

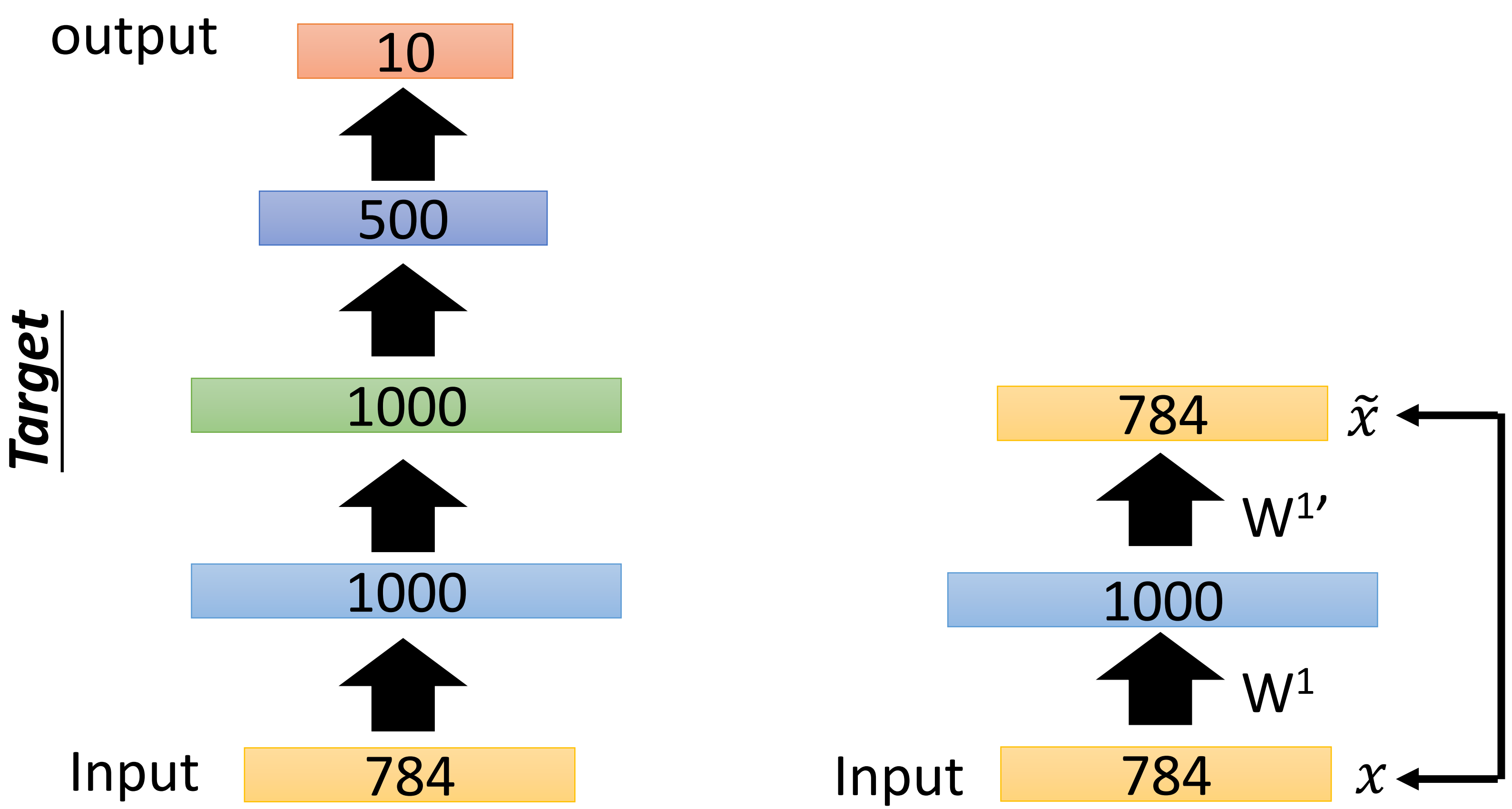
Retrieved using Euclidean distance in pixel intensity space



retrieved using 256 codes

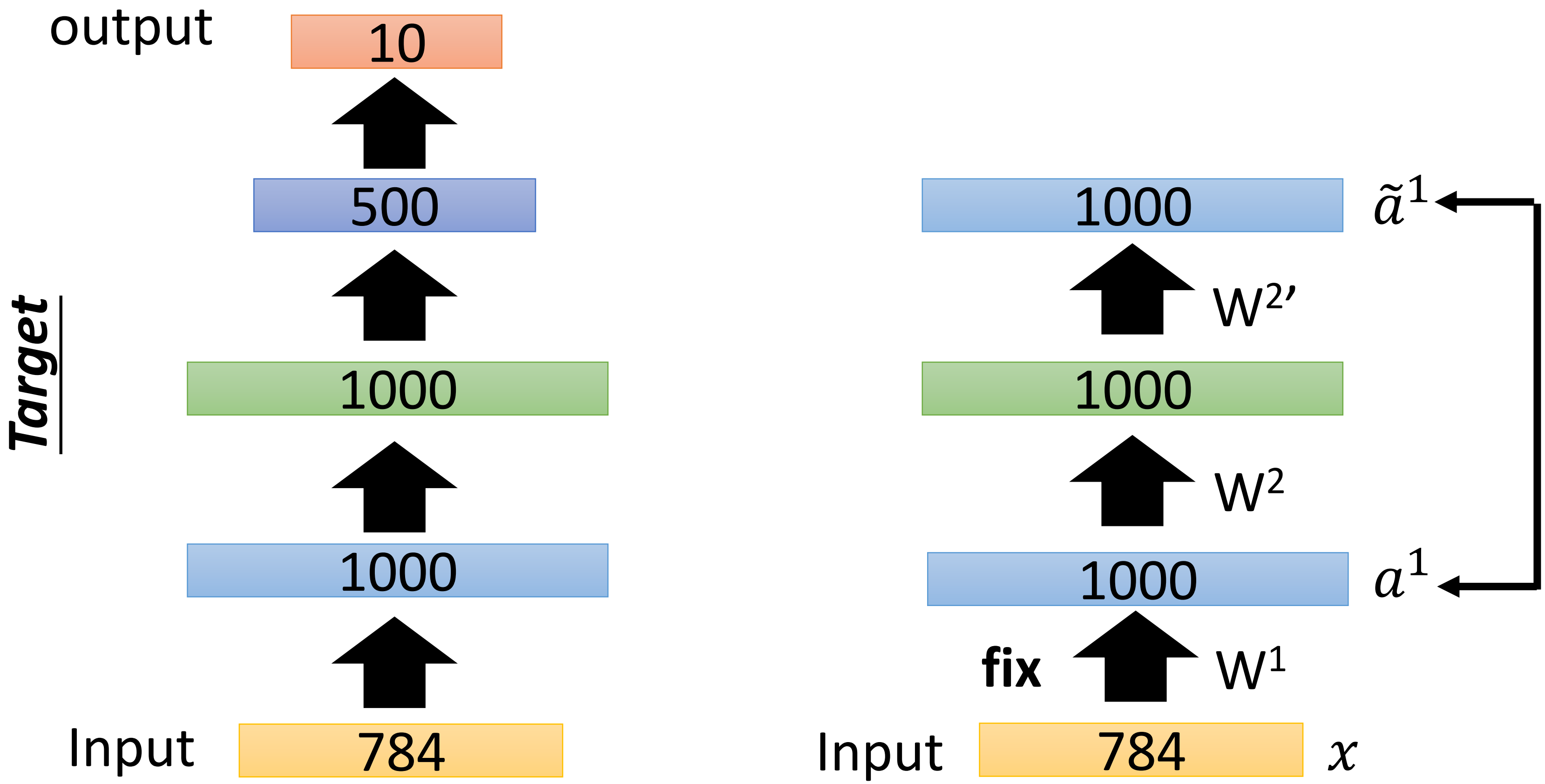
Auto-encoder – Pre-training DNN

- Greedy Layer-wise Pre-training *again*



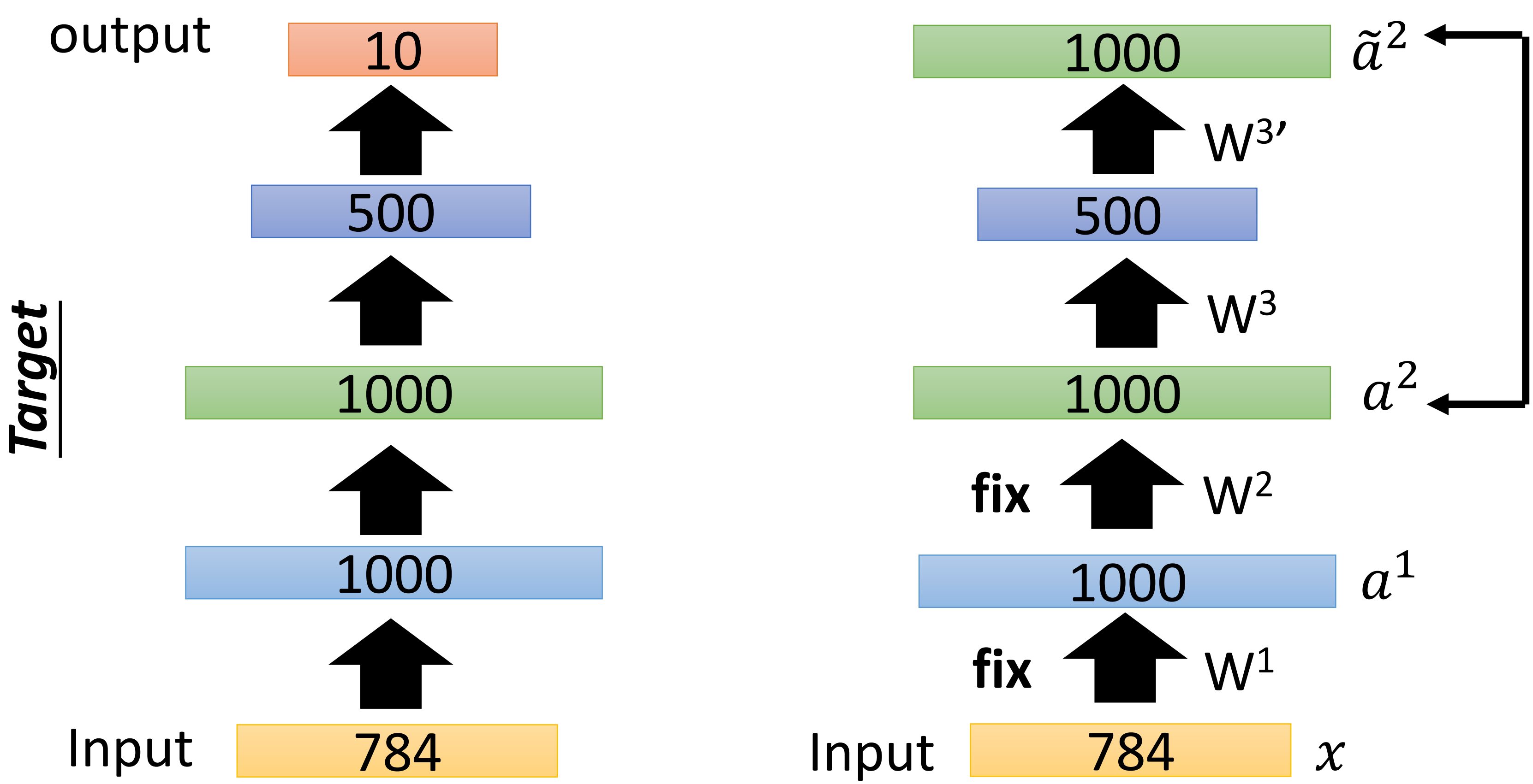
Auto-encoder – Pre-training DNN

- Greedy Layer-wise Pre-training *again*



Auto-encoder – Pre-training DNN

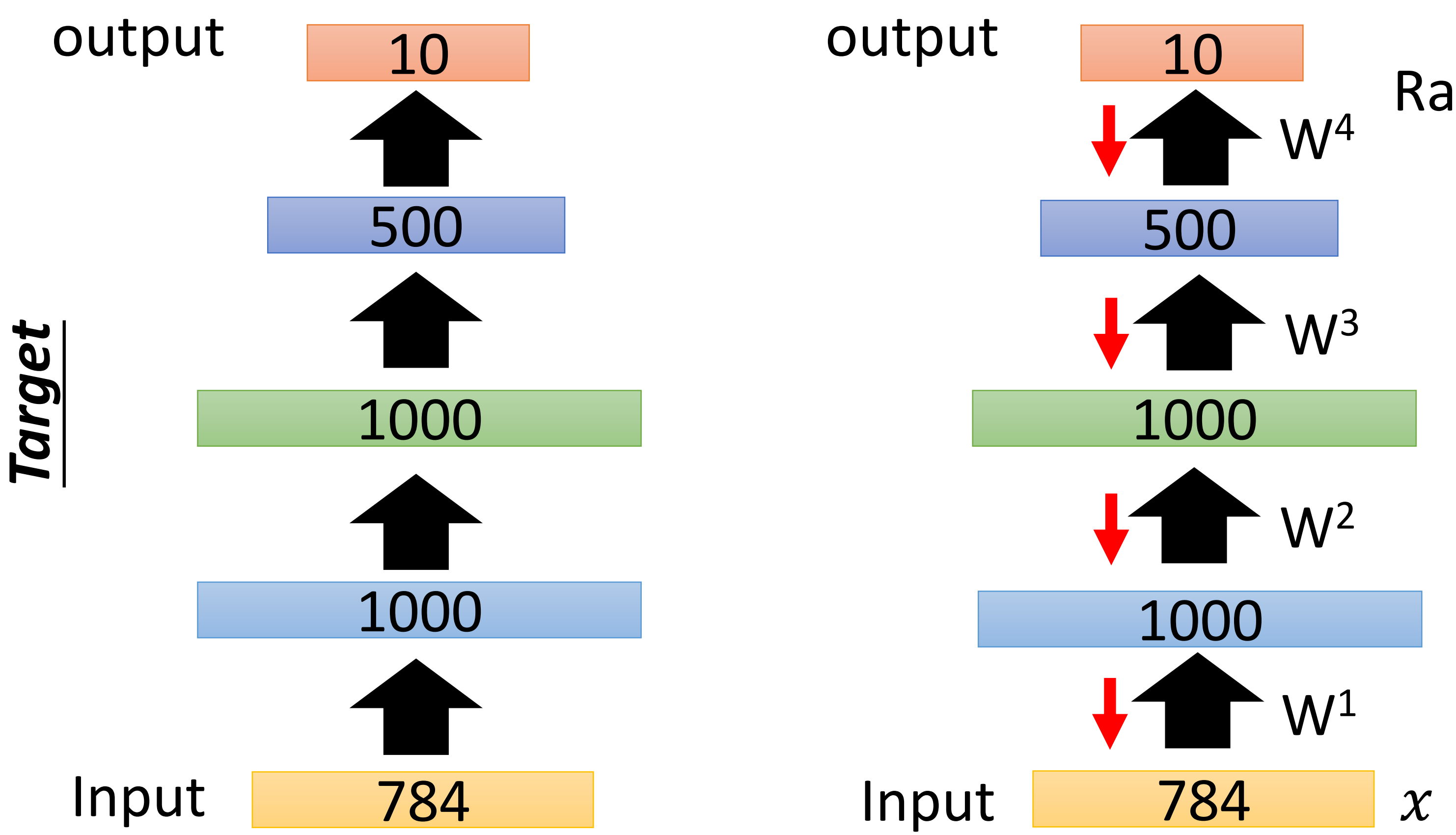
- Greedy Layer-wise Pre-training *again*



Auto-encoder – Pre-training DNN

- Greedy Layer-wise Pre-training *again*

Find-tune by backpropagation



Concluding Remarks

Concluding Remarks

- Labeling data is expensive, but it is relatively easy to collect lots of unlabeled data.
- RBM and auto-encoder exploit the unlabeled data
- RBM and auto-encoder had been popular for pre-training DNN before.
- With sufficient labelled data and ReLU, pre-training is not that important.
  - However, it is still useful when you have lots of unlabeled data but little labelled data

Plan

- 1/1 (五): 元旦放假
- 1/8 (五):
  - 2:30 ~ 3:30: TensorFlow: next generation of deep learning in Google (資工系 Seminar)
    - https://www.csie.ntu.edu.tw/app/news.php?Sn=10358
  - 4:00~: Attention-based Model
  - 23:59: presentation team decided
- 1/13 (三) 23:59: Presentation slides deadline
- 1/15 (五)
  - 14:20~: Presentation, 返鄉投票
- 1/16 (六) : 投票
- 1/20 (三) 23:59: Report deadline

Merry Christmas

&

Happy New Year

Acknowledgement

- 感謝 呂慶輝 同學指出投影片 上的錯誤