Course ID 80250993 Dates: 9/16-12/23/2021 @ I-205
Tencent/VooV Meeting 521 4678 6257 (passwd 1205)

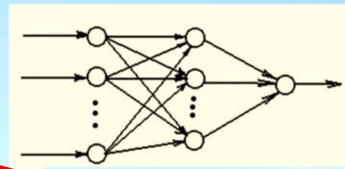
Chapter 16
Unsupervised Learning Neural Networks
(part 2)

Xuegong Zhang
Dec. 2, 2021

16.2
Hopfield Network

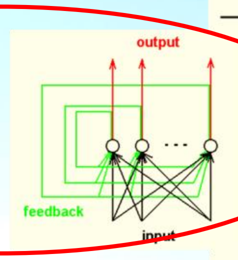
Three Major Types of ANN (in 1980s)

- Feedforward NN
  - Multi-Layer Perceptron



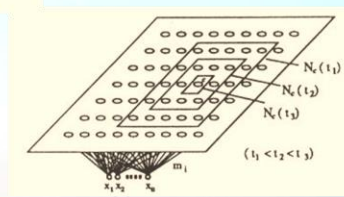
D

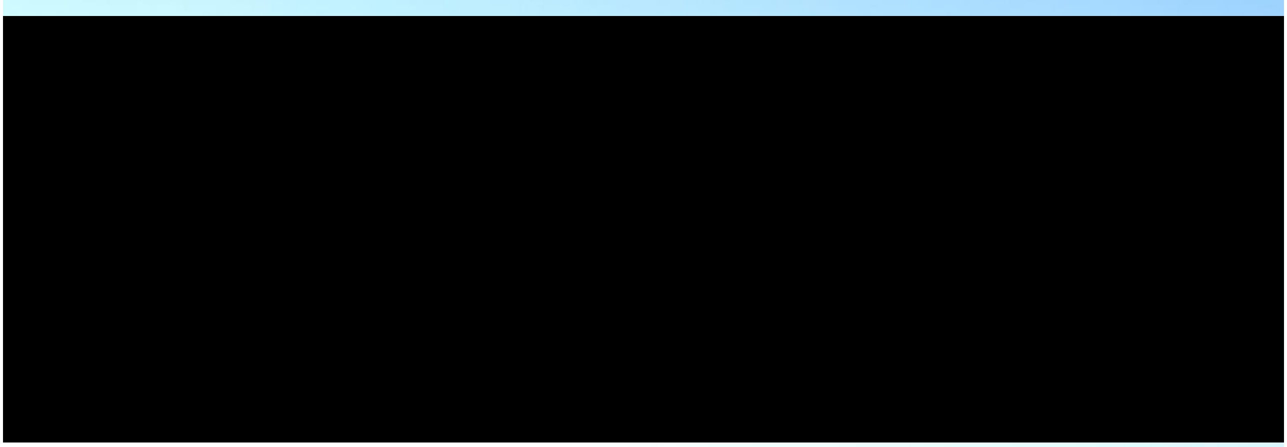
- Feedback NN
  - Hopfield NN



Energy-based neural network

- Competitive Learning NN
  - Self-organizing map

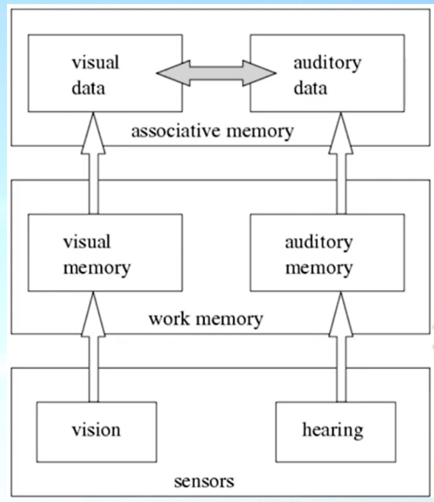




Associate Memory

- Associate the part with the whole

- Associate one type of memory with another type



Hopfield Network

- First by Little in 1974 and popularized in 1982 by John Hopfield
  - W.A. Little, The existence of persistent states in the brain, *Mathematical Biosciences* 19: 101-120, 1974
  - John J. Hopfield, Neural networks and physical systems with emergent collective computational abilities, *PNAS* 79: 2554-2558, 1982
- For "associative memory" or "content-addressable memory (CAM)"

MATHEMATICAL BIOSCIENCES **19**, 101–120 (1974)

The Existence of Persistent States in the Brain

W. A. LITTLE
Department of Physics, Stanford University, Stanford, California

Communicated by S. M. Ulam

ABSTRACT

We show that given certain plausible assumptions the existence of persistent states in a neural network can occur only if a certain transfer matrix has degenerate maximum eigenvalues. The existence of such states of persistent order is directly analogous to the existence of long range order in an Ising spin system; while the transition to the state of persistent order is analogous to the transition to the ordered phase of the spin system. It is shown that the persistent state is also characterized by correlations between neurons throughout the brain. It is suggested that these persistent states are associated with short term memory while the eigenvectors of the transfer matrix are a representation of long term memory. A numerical example is given that illustrates certain of these features.

Proc. Natl. Acad. Sci. USA
Vol. 79, pp. 2554–2558, April 1982
Biophysics

Neural networks and physical systems with emergent collective computational abilities

(associative memory/parallel processing/categorization/content-addressable memory/fail-soft devices)

J. J. HOPFIELD

ABSTRACT Computational properties of use to biological organisms or to the construction of computers can emerge as collective properties of systems having a large number of simple equivalent components (or neurons). The physical meaning of content-addressable memory is described by an appropriate phase space flow of the state of a system. A model of such a system is given, based on aspects of neurobiology but readily adapted to integrated circuits. The collective properties of this model produce a content-addressable memory which correctly yields an entire memory from any subpart of sufficient size. The algorithm for the time evolution of the state of the system is based on asynchronous parallel processing. Additional emergent collective properties include some capacity for generalization, familiarity recognition, categorization, error correction, and time sequence retention. The collective properties are only weakly sensitive to details of the modeling or the failure of individual devices.

John J. Hopfield
(July 15, 1933 -)

Biology is a Dynamic System

- John J. Hopfield:

"… If you look at the feedforward artificial neural nets, they are done as: first of all I spend some time learning, not performing, then turn off learning and I perform.

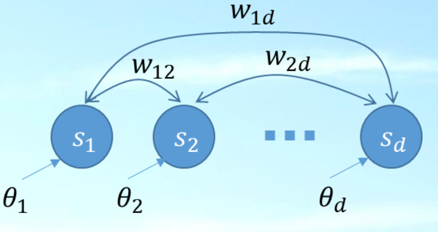
That's NOT biology.

… As I look more deeply at neurobiology, even as associate memory, I got to face the fact that the dynamics of synapse changes is going on all the time, …… "

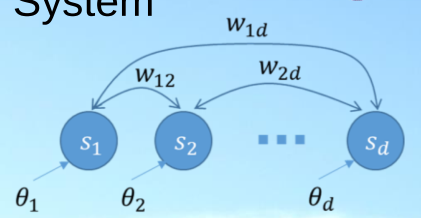
Hopfield Net as a Dynamic System

- An array of **binary threshold units** (McCulloch-Pitts neurons) fully connected with **symmetric weights**
- Each binary "configuration" of the network has an energy
- Energy (Lyapunov function)

$$E = -\frac{1}{2}\sum_{i,j} w_{ij}s_i s_j + \sum_i \theta_i s_i$$



Hopfield Net as a Dynamic System



- Min Energy

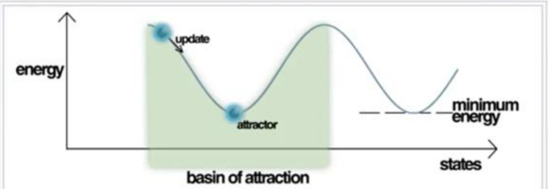
$$E = -\frac{1}{2}\sum_{i,j} w_{ij}s_i s_j + \sum_i \theta_i s_i$$

- For each node

$$\Delta E_i = E(s_i = 0) - E(s_i = 1) = \sum_j w_{ij}s_j - \theta_i$$ —— the energy gap

If $\Delta E_i \geq 0$, $s_i = 1$ lowers the energy; otherwise $s_i = 0$ lowers the energy.

➢ The "binary threshold decision rule" makes the network to settle to a minimum of energy function



Energy Landscape of a Hopfield Network, highlighting the current state of the network (up the hill), an attractor state to which it will eventually converge, a minimum energy level and a basin of attraction shaded in green. Note how the update of the Hopfield Network is always going down in Energy.

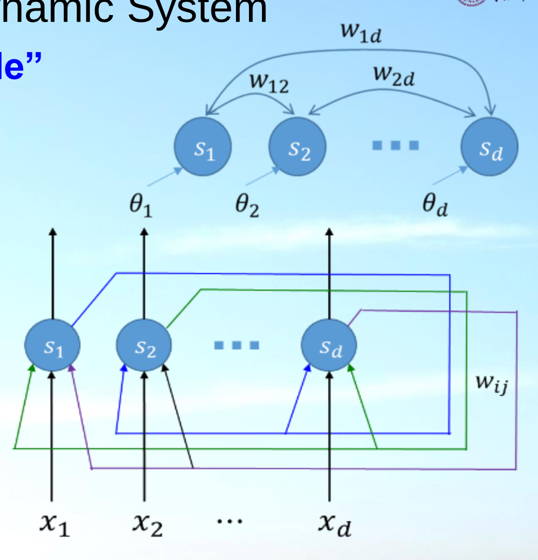
Hopfield Net as a Dynamic System

- **The "binary threshold decision rule"**
- At time 0, $\boldsymbol{s} = [s_1, \cdots, s_d]^{\mathrm{T}} = \boldsymbol{x}$,

$$\boldsymbol{x} = [x_1, \cdots, x_d]^{\mathrm{T}} \in \{1, -1\}^d$$

- $s_i(t) = f_w(s_i(t-1))$, $s_i(0) = x_i$
- At time t, updating:

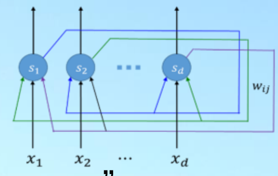
$$s_i = \begin{cases} +1 & \text{if } \sum_j w_{ij}s_j \geq \theta_i \\ -1 & \text{else} \end{cases}$$

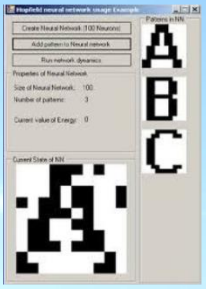


Hopfield Network

- Min Energy (Lyapunov function)

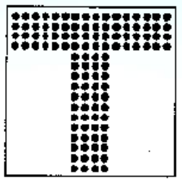
$$E = -\frac{1}{2}\sum_{i,j} w_{ij} s_i s_j + \sum_i \theta_i s_i$$



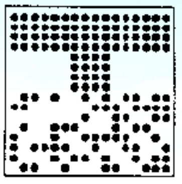


- The dynamic procedure converges to a "memory".
- Memory ←→ energy minima of neural net

Associative memory



Original 'T'



half of image corrupted by noise

"It's like reconstructing a dinosaur from a few bones."
—— Hinton

--- A good tutorial: http://web.cs.ucla.edu/~rosen/161/notes/hopfield.html

Hopfield Network

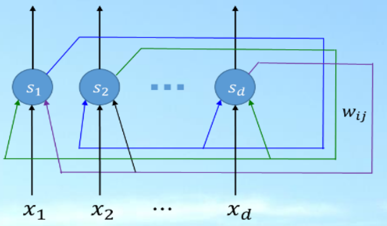
- **How to store the memory in?**
- Learning Algorithm:
  - Hebb rule: "Neurons that fire together wire together"

if $x_i, x_j \in \{-1,1\}$,

$$w_{ij} = \frac{1}{n}\sum_{k=1}^{n} x_i^k x_j^k$$

if $x_i, x_j \in \{0,1\}$,

$$w_{ij} = \frac{1}{n}\sum_{k=1}^{n} (2x_i^k - 1)(2x_j^k - 1)$$



Hebb's Rule 1949

"When an axon of cell A . . . excites cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells so that A's efficiency as one of the cells firing B is increased."
The Organization of Behavior: A Neuropsychological Theory

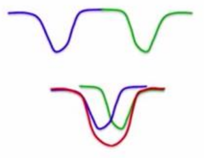
--- A good tutorial: http://web.cs.ucla.edu/~rosen/161/notes/hopfield.html

The storage capacity of a Hopfield Net

- Using Hopfield's storage rule the capacity of a totally connected net with N units is only about 0.15N memories.
  - At N bits per memory this is only $0.15\,N^2$ bits.
  - This does not make efficient use of the bits required to store the weights.
- The net has N^2 weights and biases.
- After storing M memories, each connection weight has an integer value in the range [–M, M].
- So the number of bits required to store the weights and biases is: $N^2 \log(2M+1)$

Spurious minima limit capacity

- Each time we memorize a configuration, we hope to create a new energy minimum.
  - But what if two nearby minima merge to create a minimum at an intermediate location?
  - This limits the capacity of a Hopfield net.



The state space is the corners of a hypercube. Showing it as a 1-D continuous space is a misrepresentation.

Avoiding spurious minima by unlearning

- Hopfield, Feinstein and Palmer suggested the following strategy:
  - Let the net settle from a random initial state and then do unlearning.
  - This will get rid of deep, spurious minima and increase memory capacity.
- They showed that this worked.
  - But they had no analysis.
- Crick and Mitchison proposed unlearning as a model of what dreams are for.
  - That's why you don't remember them (unless you wake up during the dream)
- But how much unlearning should we do?
  - Can we derive unlearning as the right way to minimize some cost function?

—— Slide from Hinton's lecture

Xuegong Zhang 13

10-second break

Xuegong Zhang 14

16.3
Restrictive Boltzmann Machines

Restricted Boltzmann Machines (RBM)

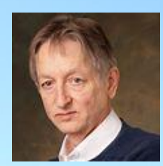
COGNITIVE SCIENCE 9, 147–169 (1985)

A Learning Algorithm for Boltzmann Machines*

DAVID H. ACKLEY
GEOFFREY E. HINTON
Computer Science Department
Carnegie-Mellon University
TERRENCE J. SEJNOWSKI
Biophysics Department
The Johns Hopkins University

The computational power of massively parallel networks of simple processing elements resides in the communication bandwidth provided by the hardware connections between elements. These connections can allow a significant

red SHG
ce in Fig.
we find
the noise
l closely
it power
emission
ngle with
ons from
Rs (see
detuning
velength
he SHG
For ex-
vertical
d a small
xcitation
incident
gnificant
gain po-



Reducing the Dimensionality of Data with Neural Networks

G. E. Hinton* and R. R. Salakhutdinov

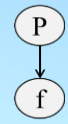
High-dimensional data can be converted to low-dimensional codes by training a multilayer neural network with a small central layer to reconstruct high-dimensional input vectors. Gradient descent can be used for fine-tuning the weights in such "autoencoder" networks, but this works well only if the initial weights are close to a good solution. We describe an effective way of initializing the weights that allows deep autoencoder networks to learn low-dimensional codes that work much better than principal components analysis as a tool to reduce the dimensionality of data.

Dimensionality reduction facilitates the classification, visualization, communication, and storage of high-dimensional data. A simple and widely used method is principal components analysis (PCA), which finds the directions of greatest variance in the data set and represents each data point by its coordinates along each of these directions. We describe a nonlinear generalization of PCA that uses an adaptive, multilayer "encoder" network

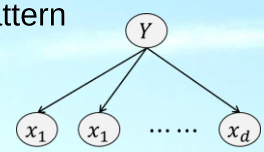
28 JULY 2006 VOL 313 SCIENCE www.sciencemag.org

What are “Patterns”?

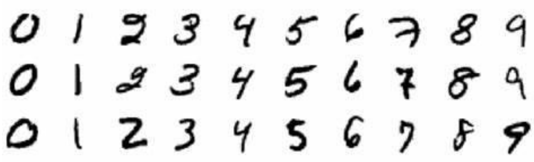
- Simple patterns
  - Single feature: distribution → pattern



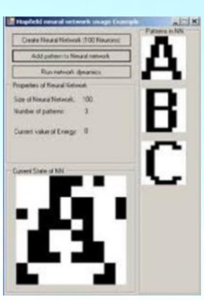
  - Multiple independent features: distributions → pattern



- Complex patterns
  - Multiple related features:
    relations among elements/features → pattern

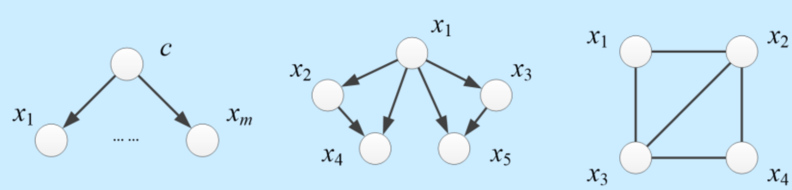


Question: How can we represent/store relations among elements?

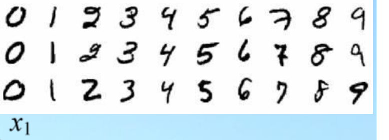


Probabilistic Graphical Models

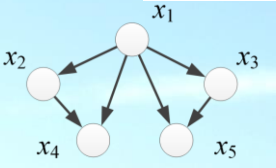
- Present the joint distribution of a set of random variables using a graph
  - Node: random variable
  - Edge: conditional dependence
  - Non-edge: conditional independence
- Applications:
  - computer vision, speech recognition
  - gene regulatory network inference
  - …

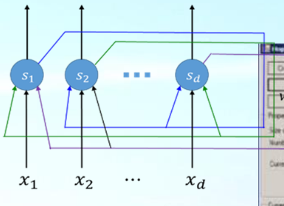


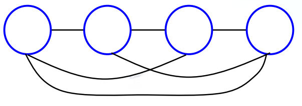
What are “Patterns”?



- How can we capture relations among features?
  - Questions (to be thought about during the final review):
    - How does MLP capture relations among pixels?
    - How does SVM capture relations among pixels?
    - How does CNN capture relations among pixels?
  - Knowledge → model → Bayesian net
  - How does Hopfield Net capture relations?
    - Hebb-rule training: finding associations among pixels
  - Can we build a full joint probabilistic model?





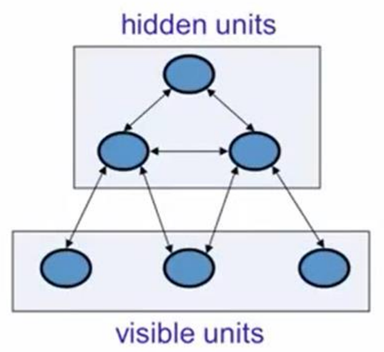


16.3.1
Boltzmann Machines

What are "Patterns"?

A different computational role for Hopfield nets

- Instead of using the net to store memories, use it to construct interpretations of sensory input.
  - The input is represented by the visible units.
  - The interpretation is represented by the states of the hidden units.
  - The badness of the interpretation is represented by the energy.

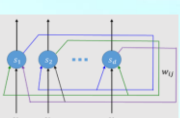


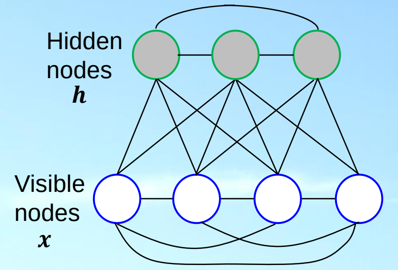
—— Slide from Hinton's lecture

Boltzmann Machines (G.E. Hinton et al, 1983)

- Binary features (0 or 1 pixels) → visible nodes
  - Binary vector $\boldsymbol{x}$
- Underlying relations → binary hidden nodes
  - Binary vector $\boldsymbol{h}$
- State of the system: $\{\boldsymbol{x}, \boldsymbol{h}\}$
- Recall
  - Min Energy (Lyapunov function)

$$E = -\frac{1}{2}\sum_{i,j} w_{ij} s_i s_j + \sum_i \theta_i s_i$$





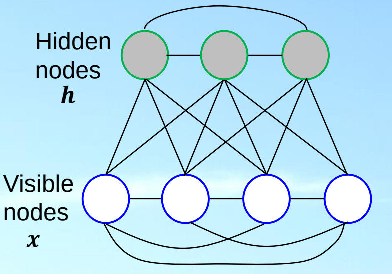
- What is a proper energy for Boltzmann Machines?

Boltzmann Machines (G.E. Hinton et al, 1983)

- A joint configuration of the network: $(\boldsymbol{x}, \boldsymbol{h})$
- Two equivalent views:
  - Define the probability of a network configuration as

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

  - Define the probability as the probability of finding the network in that configuration after the stochastic binary units are updated many times



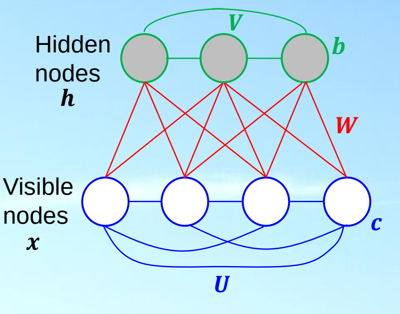
Boltzmann Machines (G.E. Hinton et al, 1983)

- Define the energy function as

$$E(\boldsymbol{x}, \boldsymbol{h}) = -\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} - \boldsymbol{b}^{\mathrm{T}}\boldsymbol{h} - \boldsymbol{h}^{\mathrm{T}}\boldsymbol{W}\boldsymbol{x} - \boldsymbol{x}^{\mathrm{T}}\boldsymbol{U}\boldsymbol{x} - \boldsymbol{h}^{\mathrm{T}}\boldsymbol{V}\boldsymbol{h}$$

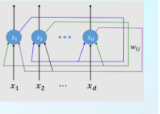
Offsets

Connection Weights



- Min Energy (Lyapunov function)

$$E = -\frac{1}{2}\sum_{i,j} w_{ij} s_i s_j + \sum_i \theta_i s_i$$



Boltzmann Machines (G.E. Hinton et al, 1983)

- The probability of a joint configuration

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

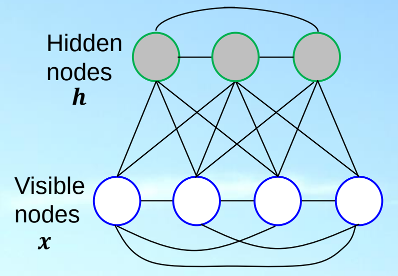
$$p(\boldsymbol{x},\boldsymbol{h}) = \frac{e^{-E(\boldsymbol{x},\boldsymbol{h})}}{\sum_{\boldsymbol{x},\boldsymbol{h}} e^{-E(\boldsymbol{x},\boldsymbol{h})}}$$

- The probability of a configuration of visible nodes $\boldsymbol{x}$

$$p(\boldsymbol{x}) = \frac{e^{-E(\boldsymbol{x})}}{Z} = \frac{\sum_{\boldsymbol{h}} e^{-E(\boldsymbol{x},\boldsymbol{h})}}{\sum_{\boldsymbol{x},\boldsymbol{h}} e^{-E(\boldsymbol{x},\boldsymbol{h})}}$$

- Normalization factor: the partition function

$$Z = \sum_{\boldsymbol{x}} e^{-E(\boldsymbol{x})} = \sum_{\boldsymbol{x},\boldsymbol{h}} e^{-E(\boldsymbol{x},\boldsymbol{h})}$$



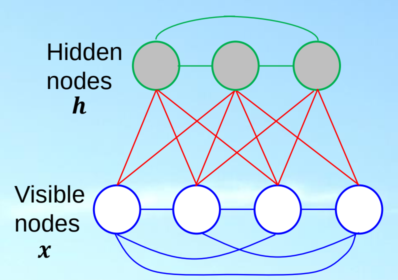
—— The Boltzmann distribution

Boltzmann Machines (G.E. Hinton et al, 1983)

- Energy with $\boldsymbol{x}$: consider all possible hidden vectors $\boldsymbol{h}$

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

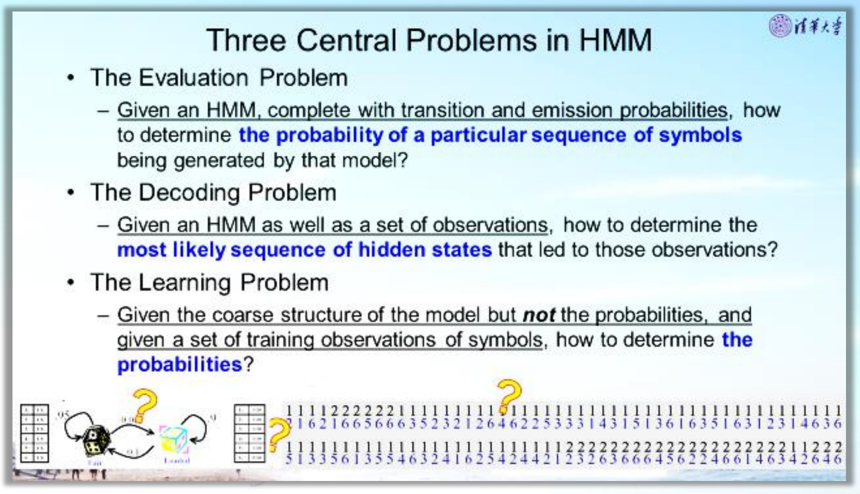
$$E(\boldsymbol{x},\boldsymbol{h}) = -\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} - \boldsymbol{b}^{\mathrm{T}}\boldsymbol{h} - \boldsymbol{h}^{\mathrm{T}}\boldsymbol{W}\boldsymbol{x} - \boldsymbol{x}^{\mathrm{T}}\boldsymbol{U}\boldsymbol{x} - \boldsymbol{h}^{\mathrm{T}}\boldsymbol{V}\boldsymbol{h}$$



➔ A fully-connected Boltzmann machine

- Can represent complex probabilistic relations
- But parameter estimation is hard

Recall: what do we do for a model

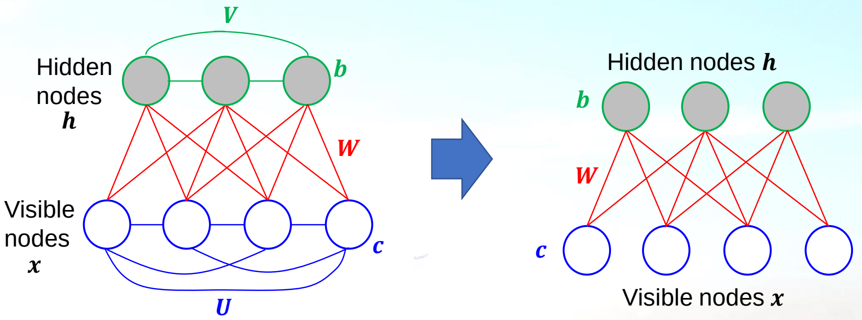


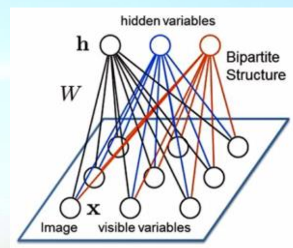
16.3.2
Restrictive Boltzmann Machines (RBM)

Restricted Boltzmann Machines (RBM)

Harmony Network or *Harmoniums* (Smolensky, 1986)

- "Restricted" connections:
  - Restricted to connections between visible and hidden nodes
  - Bipartite graph
  - Visible nodes $\boldsymbol{x} \in \{0,1\}^D$, Hidden nodes $\boldsymbol{h} \in \{0,1\}^H$





Restricted Boltzmann Machines (RBM)

Harmony Network or *Harmoniums* (Smolensky, 1986)

- Energy function

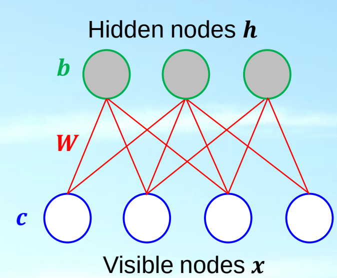
$$E(\boldsymbol{x},\boldsymbol{h}) = -\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} - \boldsymbol{b}^{\mathrm{T}}\boldsymbol{h} - \boldsymbol{h}^{\mathrm{T}}\boldsymbol{W}\boldsymbol{x}$$

- Joint probability

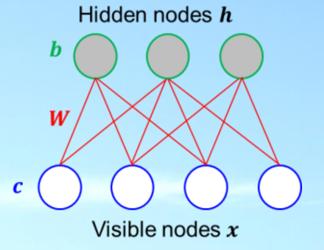
$$p(\boldsymbol{x},\boldsymbol{h}) = e^{-E(\boldsymbol{x},\boldsymbol{h})}/Z$$

- Normalization factor (partition function)

$$Z = \sum_{\boldsymbol{x},\boldsymbol{h}} e^{-E(\boldsymbol{x},\boldsymbol{h})}$$



Restricted Boltzmann Machines (RBM)



- Energy function

$$E(\boldsymbol{x},\boldsymbol{h}) = -\boldsymbol{h}^{\mathrm{T}}\boldsymbol{W}\boldsymbol{x} - \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} - \boldsymbol{b}^{\mathrm{T}}\boldsymbol{h}$$

$$= -\sum_j\sum_k W_{j,k}h_jx_k - \sum_k c_kx_k - \sum_j b_jh_j$$

- Joint probability

$$p(\boldsymbol{x},\boldsymbol{h}) = e^{-E(\boldsymbol{x},\boldsymbol{h})}/Z$$

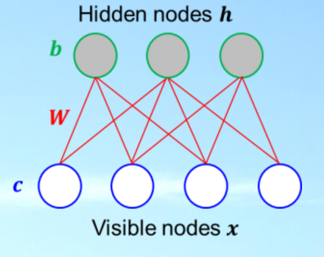
Correction in (12-23) on p.305 of the textbook

$$= \frac{1}{Z}\exp\left(\sum_j\sum_k \boldsymbol{W}_{j,k}h_jx_k + \sum_k c_kx_k + \sum_j b_jh_j\right)$$

$$= \frac{1}{Z}\prod_j\prod_k \exp(W_{j,k}h_jx_k)\prod_k \exp(c_kx_k)\prod_j \exp(b_jh_j)$$

—— *Product of Experts*

Restricted Boltzmann Machines (RBM)



$$E(\boldsymbol{x},\boldsymbol{h}) = -\boldsymbol{h}^{\mathrm{T}}\boldsymbol{W}\boldsymbol{x} - \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} - \boldsymbol{b}^{\mathrm{T}}\boldsymbol{h}$$

$$= -\sum_j\sum_k W_{j,k}h_jx_k - \sum_k c_kx_k - \sum_j b_jh_j$$

$$p(\boldsymbol{x},\boldsymbol{h}) = e^{-E(\boldsymbol{x},\boldsymbol{h})}/Z$$

$$= \frac{1}{Z}\prod_j\prod_k \exp(W_{j,k}h_jx_k)\prod_k \exp(c_kx_k)\prod_j \exp(b_jh_j)$$

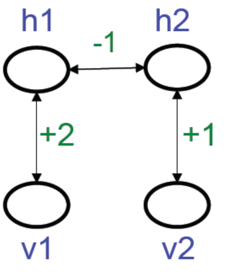
$$p(\boldsymbol{x}) = \frac{1}{Z}e^{-E(\boldsymbol{x})} = \frac{1}{Z}\sum_{\boldsymbol{h}} e^{-E(\boldsymbol{x},\boldsymbol{h})}$$

- Conditional probability

$$p(\boldsymbol{h}|\boldsymbol{x}) = \frac{p(\boldsymbol{x},\boldsymbol{h})}{p(\boldsymbol{x})} = \frac{1}{Z'}\prod_j \exp(b_jh_j + h_j\boldsymbol{W}_{j\cdot}\boldsymbol{x})$$

$\mathbf{v}$	$\mathbf{h}$	$-E$	e^{-E}	$p(\mathbf{v},\mathbf{h})$	$p(\mathbf{v})$
1 1	1 1	2	7.39	.186	
1 1	1 0	2	7.39	.186	
1 1	0 1	1	2.72	.069	0.466
1 1	0 0	0	1	.025	
1 0	1 1	1	2.72	.069	
1 0	1 0	2	7.39	.186	
1 0	0 1	0	1	.025	0.305
1 0	0 0	0	1	.025	
0 1	1 1	0	1	.025	
0 1	1 0	0	1	.025	
0 1	0 1	1	2.72	.069	0.144
0 1	0 0	0	1	.025	
0 0	1 1	-1	0.37	.009	
0 0	1 0	0	1	.025	
0 0	0 1	0	1	.025	0.084
0 0	0 0	0	1	.025	
			39.70		

An example of how weights define a distribution



—— Slide from Hinton's lecture

Restricted Boltzmann Machines (RBM)

- Conditional probability

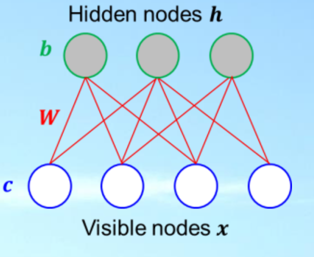
$$p(\boldsymbol{h}|\boldsymbol{x}) = \frac{p(\boldsymbol{x},\boldsymbol{h})}{p(\boldsymbol{x})} = \frac{1}{Z'}\prod_j \exp(b_j h_j + h_j \boldsymbol{W}_{j\cdot}\boldsymbol{x})$$

$$P(h_j = 1|\boldsymbol{x}) = \frac{\exp(b_j + \boldsymbol{W}_{j\cdot}\boldsymbol{x})}{1 + \exp(b_j + \boldsymbol{W}_{j\cdot}\boldsymbol{x})} = \mathrm{sigm}(b_j + \boldsymbol{W}_{j\cdot}\boldsymbol{x})$$

$$P(h_j = 0|\boldsymbol{x}) = 1 - \mathrm{sigm}(b_j + \boldsymbol{W}_{j\cdot}\boldsymbol{x}) = \mathrm{sigm}(-b_j - \boldsymbol{W}_{j\cdot}\boldsymbol{x})$$

$$p(\boldsymbol{h}|\boldsymbol{x}) = \prod_j p(h_j|\boldsymbol{x}) = \prod_j \mathrm{sigm}\big((2h_j - 1)(b_j + \boldsymbol{W}_{j\cdot}\boldsymbol{x})\big)$$

$$p(\boldsymbol{x}|\boldsymbol{h}) = \prod_k p(x_k|\boldsymbol{h}) = \prod_k \mathrm{sigm}((2x_k - 1)(c_k + \boldsymbol{h}\boldsymbol{W}_{\cdot k}))$$

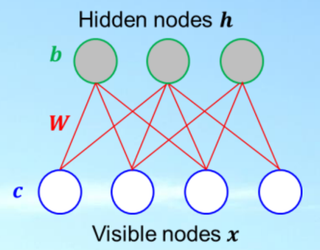


sigm(): sigmoid function

Restricted Boltzmann Machines (RBM)

$$p(\boldsymbol{h}|\boldsymbol{x}) = \prod_j \mathrm{sigm}\big((2h_j - 1)(b_j + \boldsymbol{W}_{j\cdot}\boldsymbol{x})\big)$$

$$p(\boldsymbol{x}|\boldsymbol{h}) = \prod_k \mathrm{sigm}\big((2x_k - 1)(c_k + \boldsymbol{h}\boldsymbol{W}_{\cdot k})\big)$$



- A generative model
  - Knowing the model and data, we can sample the hidden states
  - Giving the model and hidden states, we can sample the data

单选题 1分

设置

4-min break

A Yes, I’m back to my computer.

B Sorry, not yet.

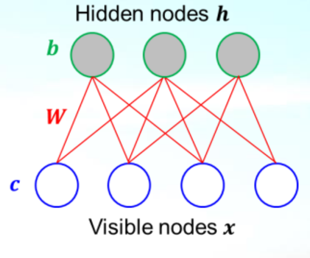
提交

16.3.3
Learning of RBM

The Learning Problem

- Knowing the model and data, we can sample the hidden states.
- Giving the model and hidden states, we can sample the data.
- **Giving the data, how can we learn the model (weights, offsets)?**
    - ➢ Q: What is the basic idea for parametric learning?

D



Learning the parameters of RBM

$$p(\boldsymbol{x},\boldsymbol{h}) = e^{-E(\boldsymbol{x},\boldsymbol{h})}/Z = \frac{1}{Z}\prod_j\prod_k \exp(W_{j,k}h_j x_k)\prod_k \exp(c_k x_k)\prod_j \exp(b_j h_j)$$

Likelihood: $P(\boldsymbol{x};\boldsymbol{\theta}) = \sum_{\boldsymbol{h}} p(\boldsymbol{x},\boldsymbol{h}) = \sum_{\boldsymbol{h}} e^{-E(\boldsymbol{x},\boldsymbol{h})}/Z$

The general idea of EM algorithms

- A statistical model for X with unobserved latent data or missing values Z and a vector of unknown parameters θ.

$$p(X,Z;\theta)$$

- Likelihood of the observed data

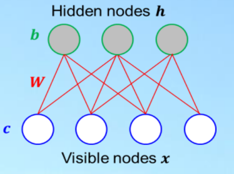
$$L(\theta;X) = p(X|\theta) = \int p(X,Z|\theta)dZ$$

- However, we don't know Z and enumerating all possibilities is often infeasible.
- The idea is to make the estimation in two iterative steps:
  - Get the expected value of the log likelihood of θ based on some estimate of Z given X with the current estimation $\theta^{(t)}$ Expectation (E-step)
  - Find the next estimation $\theta^{(t+1)}$ that maximizes this expected log likelihood Maximization (M-step)

D

39

Learning Procedures for RBM



- MLE (Maximum Likelihood Estimation):

$$\max_{\theta} P(\boldsymbol{x};\theta)$$

$$P(\boldsymbol{x};\theta) = \sum_{\boldsymbol{h}} p(\boldsymbol{x},\boldsymbol{h}) = \sum_{\boldsymbol{h}} e^{-E(\boldsymbol{x},\boldsymbol{h})}/Z\ , \qquad Z = \sum_{\boldsymbol{x},\boldsymbol{h}} e^{-E(\boldsymbol{x},\boldsymbol{h})}$$

- Minimum Loss: $\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}(-\log P(\boldsymbol{x}))$

$$\frac{\partial(-\log P(\boldsymbol{x}))}{\partial\boldsymbol{\theta}} = \frac{\partial\left(-\log\sum_{\boldsymbol{h}} e^{-E(\boldsymbol{x},\boldsymbol{h})} + \log\sum_{\boldsymbol{x},\boldsymbol{h}} e^{-E(\boldsymbol{x},\boldsymbol{h})}\right)}{\partial\boldsymbol{\theta}}$$

40

Learning Procedures for RBM

- Minimum Loss: $\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}(-\log P(\boldsymbol{x}))$

$$P(\boldsymbol{x}) = \sum_{\boldsymbol{h}} e^{-E(\boldsymbol{x},\boldsymbol{h})} / \sum_{\boldsymbol{x},\boldsymbol{h}} e^{-E(\boldsymbol{x},\boldsymbol{h})}$$

- Gradient descent: Updating weights along the negative direction of the gradient $\frac{\partial(-\log P(\boldsymbol{x}))}{\partial \boldsymbol{\theta}}$

$$\frac{\partial(-\log P(\boldsymbol{x}))}{\partial \boldsymbol{\theta}} = \frac{\partial\left(-\log \sum_{\boldsymbol{h}} e^{-E(\boldsymbol{x},\boldsymbol{h})} + \log \sum_{\boldsymbol{x},\boldsymbol{h}} e^{-E(\boldsymbol{x},\boldsymbol{h})}\right)}{\partial \boldsymbol{\theta}}$$

$$= \frac{1}{\sum_{\boldsymbol{h}} e^{-E(\boldsymbol{x},\boldsymbol{h})}} \sum_{h} e^{-E(\boldsymbol{x},\boldsymbol{h})} \frac{\partial E(\boldsymbol{x},\boldsymbol{h})}{\partial \boldsymbol{\theta}} - \frac{1}{\sum_{\boldsymbol{x},\boldsymbol{h}} e^{-E(\boldsymbol{x},\boldsymbol{h})}} \sum_{x,h} e^{-E(\boldsymbol{x},\boldsymbol{h})} \frac{\partial E(\boldsymbol{x},\boldsymbol{h})}{\partial \boldsymbol{\theta}} = \sum_{\boldsymbol{h}} p(\boldsymbol{h}|\boldsymbol{x}) \frac{\partial E(\boldsymbol{x},\boldsymbol{h})}{\partial \boldsymbol{\theta}} - \sum_{\boldsymbol{x},\boldsymbol{h}} p(\boldsymbol{x},\boldsymbol{h}) \frac{\partial E(\boldsymbol{x},\boldsymbol{h})}{\partial \boldsymbol{\theta}}$$

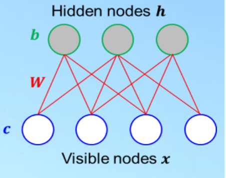
i.e. $$\frac{\partial(-\log P(\boldsymbol{x}))}{\partial \boldsymbol{\theta}} = \mathbb{E}_{P(\boldsymbol{h}|\boldsymbol{x})} \frac{\partial E(\boldsymbol{x},\boldsymbol{h})}{\partial \boldsymbol{\theta}} - \mathbb{E}_{P(\boldsymbol{x},\boldsymbol{h})} \frac{\partial E(\boldsymbol{x},\boldsymbol{h})}{\partial \boldsymbol{\theta}}$$

- To calculate the gradient, we need to sample from $P(\boldsymbol{h}|\boldsymbol{x})$ and $P(\boldsymbol{x},\boldsymbol{h})$.

Intractable

41

The Gradients



$$\frac{\partial(-\log P(\boldsymbol{x}))}{\partial \boldsymbol{\theta}} = \mathbb{E}_{P(\boldsymbol{h}|\boldsymbol{x})} \frac{\partial E(\boldsymbol{x},\boldsymbol{h})}{\partial \boldsymbol{\theta}} - \mathbb{E}_{P(\boldsymbol{x},\boldsymbol{h})} \frac{\partial E(\boldsymbol{x},\boldsymbol{h})}{\partial \boldsymbol{\theta}}$$

Expectation over observed data
Negative of the direction:
energy ↓ , likelihood ↑

Expectation over all values (random data)
Negative of the negative direction:
energy ↑, likelihood ↓

$$E(\boldsymbol{x},\boldsymbol{h}) = -\boldsymbol{h}^{\mathrm{T}}\boldsymbol{W}\boldsymbol{x} - \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} - \boldsymbol{b}^{\mathrm{T}}\boldsymbol{h} = -\sum_j \sum_k W_{j,k} h_j x_k - \sum_k c_k x_k - \sum_j b_j h_j$$

$$\frac{\partial(-\log P(\boldsymbol{x}))}{\partial \boldsymbol{W}} = -\mathbb{E}_{\boldsymbol{h}}\left(\boldsymbol{h}(\boldsymbol{x}(t))^{\mathrm{T}}\boldsymbol{x}(t)\right) + \mathbb{E}_{\boldsymbol{x},\boldsymbol{h}}\boldsymbol{h}^{\mathrm{T}}\boldsymbol{x}$$

$$\frac{\partial(-\log P(\boldsymbol{x}))}{\partial \boldsymbol{c}} = -\boldsymbol{x}(t) + \mathbb{E}_{\boldsymbol{x},\boldsymbol{h}}\boldsymbol{x}$$

$$\frac{\partial(-\log P(\boldsymbol{x}))}{\partial \boldsymbol{b}} = -\mathbb{E}_{\boldsymbol{h}}\boldsymbol{h}(\boldsymbol{x}(t)) + \mathbb{E}_{\boldsymbol{x},\boldsymbol{h}}\boldsymbol{h}$$

$\mathbb{E}_{P(\boldsymbol{x},\boldsymbol{h})}$ Intractable partition function

$\mathbb{E}_{P(\boldsymbol{h}|\boldsymbol{x})}$ $p(\boldsymbol{h}|\boldsymbol{x}) = \prod_j \operatorname{sigm}\left((2h_j - 1)(b_j + \boldsymbol{W}_{j\cdot}\boldsymbol{x})\right)$

Xuegong Zhang

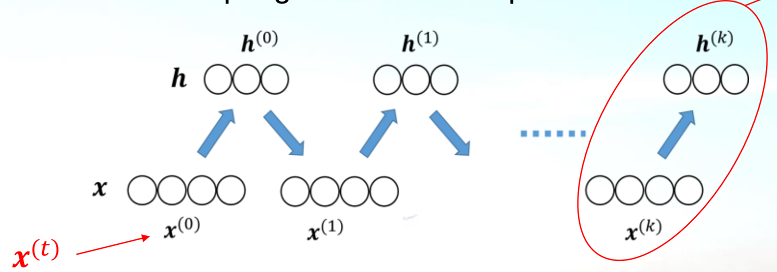
42

The Contrastive Divergence (CD) Algorithm

$$\frac{\partial(-\log P(\boldsymbol{x}))}{\partial\boldsymbol{\theta}} = \mathbb{E}_{P(\boldsymbol{h}|\boldsymbol{x})}\frac{\partial E(\boldsymbol{x},\boldsymbol{h})}{\partial\boldsymbol{\theta}} - \mathbb{E}_{P(\boldsymbol{x},\boldsymbol{h})}\frac{\partial E(\boldsymbol{x},\boldsymbol{h})}{\partial\boldsymbol{\theta}}$$

- Idea for solution
  - Replace the expectation by a point estimate at $\tilde{\boldsymbol{x}}$
  - Obtain the point $\tilde{\boldsymbol{x}}$ by Gibbs sampling
  - Start sampling chain at a sample $\boldsymbol{x}^{(t)}$

Estimation of the expectation $\mathbb{E}_{P(\boldsymbol{x},\boldsymbol{h})}(\,)$

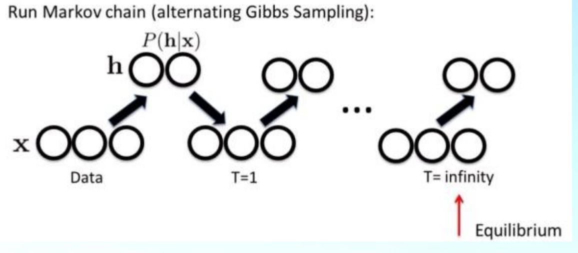


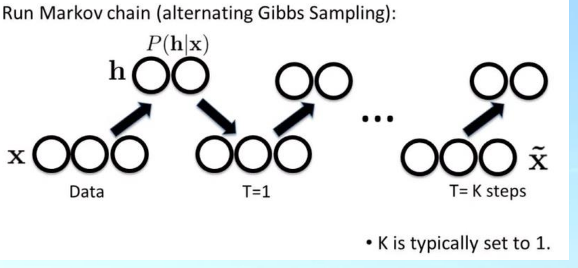
$$p(\boldsymbol{h}|\boldsymbol{x}) = \prod_j \mathrm{sigm}\big((2h_j - 1)(b_j + \boldsymbol{W}_{j\cdot}\boldsymbol{x})\big)$$

$$p(\boldsymbol{x}|\boldsymbol{h}) = \prod_k \mathrm{sigm}\big((2x_k - 1)(c_k + \boldsymbol{h}\boldsymbol{W}_{\cdot k})\big)$$

MCMC: Markov chain Monte Carlo (alternating Gibbs Sampling)

The Contrastive Divergence (CD) Algorithm





$$\frac{\partial(-\log P(\boldsymbol{x}))}{\partial\boldsymbol{W}} = -\mathbb{E}_{\boldsymbol{h}}\big(\boldsymbol{h}(\boldsymbol{x}(t))^{\mathrm{T}}\boldsymbol{x}(t)\big) + \mathbb{E}_{\boldsymbol{x},\boldsymbol{h}}\boldsymbol{h}^{\mathrm{T}}\boldsymbol{x}$$

$$\frac{\partial(-\log P(\boldsymbol{x}))}{\partial\boldsymbol{c}} = -\boldsymbol{x}(t) + \mathbb{E}_{\boldsymbol{x},\boldsymbol{h}}\boldsymbol{x}$$

$$\frac{\partial(-\log P(\boldsymbol{x}))}{\partial\boldsymbol{b}} = -\mathbb{E}_{\boldsymbol{h}}\boldsymbol{h}(\boldsymbol{x}(t)) + \mathbb{E}_{\boldsymbol{x},\boldsymbol{h}}\boldsymbol{h}$$

$$\Delta\boldsymbol{W} = -\frac{\partial(-\log P(\boldsymbol{x}))}{\partial\boldsymbol{W}} = \underline{\boldsymbol{h}\big(\boldsymbol{x}(t)\big)^{\mathrm{T}}\boldsymbol{x}(t)} - \underline{\tilde{\boldsymbol{h}}^{\mathrm{T}}\tilde{\boldsymbol{x}}}$$

$$\Delta\boldsymbol{c} = \boldsymbol{x}(t) - \tilde{\boldsymbol{x}}$$

$$\Delta\boldsymbol{b} = \boldsymbol{h}(\boldsymbol{x}(t)) - \tilde{\boldsymbol{h}}$$

fake sample & hidden state: **weaken their connections**

true sample & most probable hidden state: **strengthen their connections**

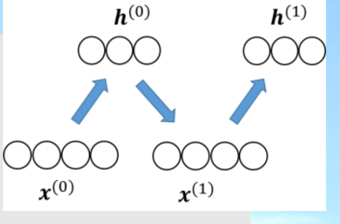
CD-k Algorithm

- For each training example $\mathbf{x}^{(t)}$
  - Generate a negative sample $\tilde{\mathbf{x}}$ using k steps of Gibbs sampling, starting at the data point $\mathbf{x}^{(t)}$
  - Update model parameters:

$$\mathbf{W} \Leftarrow \mathbf{W} + \alpha\left(\mathbf{h}(\mathbf{x}^{(t)})\,\mathbf{x}^{(t)\top} - \mathbf{h}(\tilde{\mathbf{x}})\,\tilde{\mathbf{x}}^{\top}\right)$$
$$\mathbf{b} \Leftarrow \mathbf{b} + \alpha\left(\mathbf{h}(\mathbf{x}^{(t)}) - \mathbf{h}(\tilde{\mathbf{x}})\right)$$
$$\mathbf{c} \Leftarrow \mathbf{c} + \alpha\left(\mathbf{x}^{(t)} - \tilde{\mathbf{x}}\right)$$

  - Go back to 1 until stopping criteria

CD-1 Algorithm



Set learning rate η

Parameters to be learned: weights $\boldsymbol{W}$, offset vectors $\boldsymbol{b}$ and $\boldsymbol{c}$

(1) Initialization

(2) Input a training example $\boldsymbol{x_0}$

Sampling:

a) For all hidden nodes, calculate $P(h_j = 1|\boldsymbol{x_0}) = \text{sigm}(b_j + \boldsymbol{W}_{j\cdot}\boldsymbol{x_0})$, sample $\boldsymbol{h_0}$

b) For all visible nodes, calculate $P(x_k = 1|\boldsymbol{h}_0) = \text{sigm}(c_k + \boldsymbol{h_0}\boldsymbol{W}_{\cdot k})$, sample $\boldsymbol{x_1}$

c) For all hidden nodes, calculate $P(h_j = 1|\boldsymbol{x_1}) = \text{sigm}(b_j + \boldsymbol{W}_{j\cdot}\boldsymbol{x_1})$, sample $\boldsymbol{h_1}$

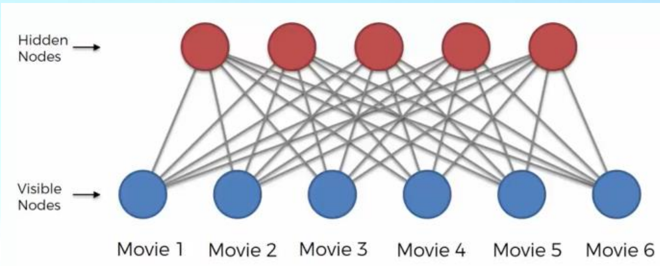
Updating:

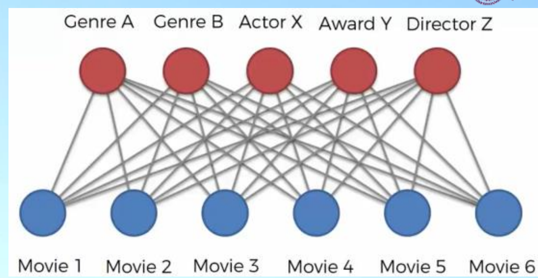
$$\boldsymbol{W} \Leftarrow \boldsymbol{W} + \eta(\boldsymbol{h}_0^{\mathrm{T}}\boldsymbol{x}_0 - \boldsymbol{h}_1^{\mathrm{T}}\boldsymbol{x}_1)$$
$$\boldsymbol{c} \Leftarrow \boldsymbol{c} + \eta(\boldsymbol{x}_0 - \boldsymbol{x}_1)$$
$$\boldsymbol{b} \Leftarrow \boldsymbol{b} + \eta(\boldsymbol{h}_0 - \boldsymbol{h}_1)$$

(3) Check for convergence: If converged or reached pre-set training rounds, stop; Otherwise go to (2).

10-second break

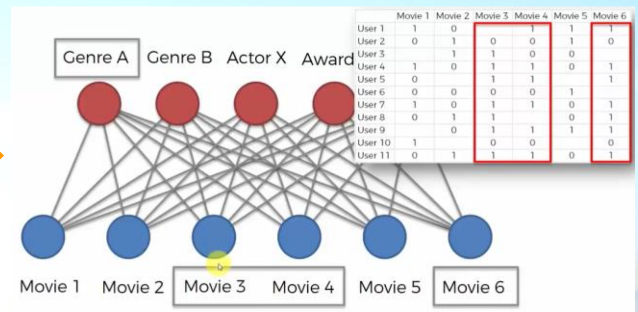
16.3.4
Using RBM for Recommendation

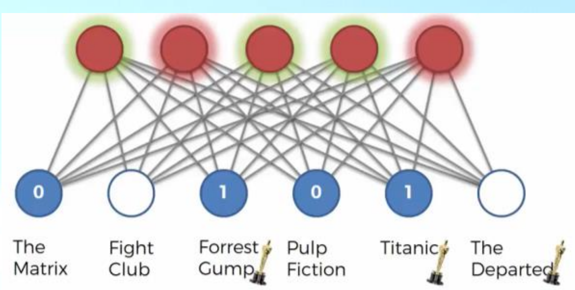




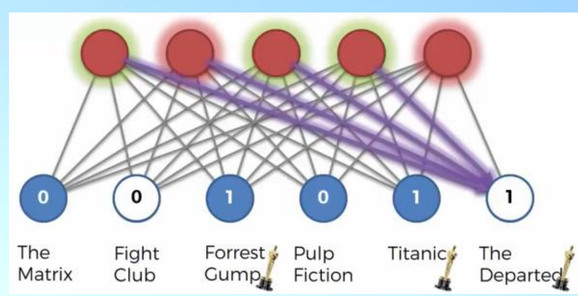
	Movie 1	Movie 2	Movie 3	Movie 4	Movie 5	Movie 6
User 1	1	0		1	1	1
User 2	0	1	0	0	1	0
User 3		1	1	0	0	
User 4	1	0	1	1	0	1
User 5	0		1	1		1
User 6	0	0	0	0	1	
User 7	1	0	1	1	0	1
User 8	0	1	1		0	1
User 9		0	1	1	1	1
User 10	1		0	0		0
User 11	0	1	1	1	0	1

Training





Prediction



16.3.5
RBM with Different Data Modalities

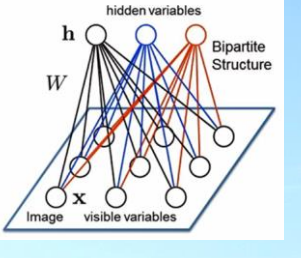
Gaussian Bernoulli RBMs

- Real-valued visible variables $\boldsymbol{x}$

$$E(\boldsymbol{x},\boldsymbol{h}) = -\boldsymbol{h}^{\mathrm{T}}\boldsymbol{W}\boldsymbol{x} - \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} - \boldsymbol{b}^{\mathrm{T}}\boldsymbol{h} + \frac{1}{2}\boldsymbol{x}^{\mathrm{T}}\boldsymbol{x}$$

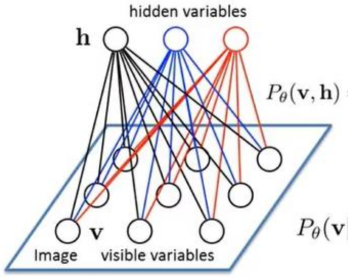
- $p(\boldsymbol{x}|\boldsymbol{h})$ becomes a Gaussian distribution:

$$p(\boldsymbol{x}|\boldsymbol{h}) \sim N(\boldsymbol{c} + \boldsymbol{W}^{\mathrm{T}}\boldsymbol{h}, \boldsymbol{I})$$



Slides from: Russ Salakhutdinov, Course 10707 at CMU ML Department
http://www.cs.cmu.edu/~rsalakhu/10707

Gaussian Bernoulli RBMs



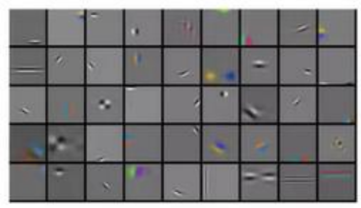
$$P_\theta(\mathbf{v},\mathbf{h}) = \frac{1}{\mathcal{Z}(\theta)}\exp\left(\overbrace{\sum_{i=1}^{D}\sum_{j=1}^{F} W_{ij}h_j\frac{v_i}{\sigma_i}}^{\text{Pair-wise}} + \overbrace{\sum_{i=1}^{D}\frac{(v_i-b_i)^2}{2\sigma_i^2} + \sum_{j=1}^{F} a_j h_j}^{\text{Unary}}\right)$$

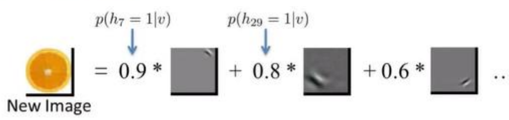
$$\theta = \{W, a, b\}$$

$$P_\theta(\mathbf{v}|\mathbf{h}) = \prod_{i=1}^{D} P_\theta(v_i|\mathbf{h}) = \prod_{i=1}^{D}\mathcal{N}\left(b_i + \sum_{j=1}^{F} W_{ij}h_j, \sigma_i^2\right)$$

4 million **unlabelled** images

Learned features (out of 10,000)





Gaussian Bernoulli RBMs

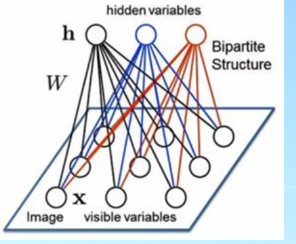
- Interpretation:

Mixture of exponential number of Gaussians

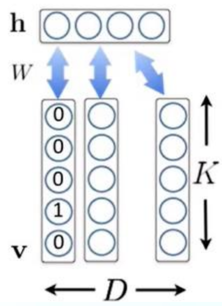
$$p(\boldsymbol{x}) = \sum_{\boldsymbol{h}} p(\boldsymbol{x}|\boldsymbol{h})P(\boldsymbol{h})$$

where $P(\boldsymbol{h}) = \int_{\boldsymbol{x}} p(\boldsymbol{x},\boldsymbol{h})d\boldsymbol{x}$

and $p(x_i|\boldsymbol{h}) = \frac{1}{\sqrt{2\pi}\sigma_i}\exp\left(-\frac{\left(x_i - b_i - \sigma_i\sum_j W_{ij}h_j\right)^2}{2\sigma_i^2}\right)$



RBMs for Word Counts



$$P_\theta(\mathbf{v},\mathbf{h}) = \frac{1}{\mathcal{Z}(\theta)}\exp\left(\overbrace{\sum_{i=1}^{D}\sum_{k=1}^{K}\sum_{j=1}^{F} W_{ij}^{k} v_i^k h_j}^{\text{Pair-wise}} + \overbrace{\sum_{i=1}^{D}\sum_{k=1}^{K} v_i^k b_i^k + \sum_{j=1}^{F} h_j a_j}^{\text{Unary}}\right)$$

$$\theta = \{W, a, b\}$$

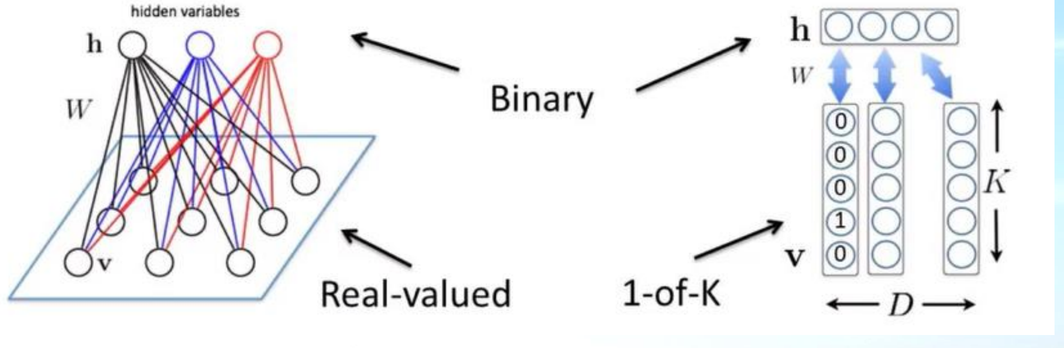
$$P_\theta(v_i^k = 1|\mathbf{h}) = \frac{\exp\left(b_i^k + \sum_{j=1}^{F} h_j W_{ij}^k\right)}{\sum_{q=1}^{K}\exp\left(b_i^q + \sum_{j=1}^{F} h_j W_{ij}^q\right)}$$

- Replicated Softmax Model: undirected topic model:
  - Stochastic 1-of-K visible variables
  - Stochastic binary hidden variables

Salakhutdinov & Hinton, NIPS 2010

Different Data Modalities

- Binary/Gaussian/Softmax RBMs: All have binary hidden variables but use them to model different kinds of data.



Building a Probabilistic Model

- Learn a joint density model: $P(\mathbf{h}, \mathbf{v}_{\text{image}}, \mathbf{v}_{\text{text}})$.
- **h**: "fused" representation for classification, retrieval.

$P(\mathbf{h}|\mathbf{v}_{\text{image}}, \mathbf{v}_{\text{text}})$

"Concept"

$\mathbf{h}$

sunset, pacificocean, bakerbeach, seashore, ocean

$\mathbf{v}_{\text{image}}$ $\mathbf{v}_{\text{text}}$

- Learn a joint density model: $P(\mathbf{h}, \mathbf{v}_{\text{image}}, \mathbf{v}_{\text{text}})$.
- **h**: "fused" representation for classification, retrieval.
- Generate data from conditional distributions for
  - Image Annotation

$P(\mathbf{h}, \mathbf{v}_{\text{text}}|\mathbf{v}_{\text{image}})$

"Concept"

$\mathbf{h}$

Missing Data

$\mathbf{v}_{\text{image}}$ $\mathbf{v}_{\text{text}}$

- Generate data from conditional distributions for
  - Image Annotation
  - Image Retrieval

$P(\mathbf{h}, \mathbf{v}_{\text{image}}|\mathbf{v}_{\text{text}})$

"Concept"

$\mathbf{h}$

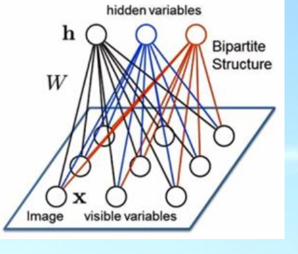
Missing Data

sunset, pacificocean, bakerbeach, seashore, ocean

$\mathbf{v}_{\text{image}}$ $\mathbf{v}_{\text{text}}$

Multiple Application Domains

- Natural Images
- Texts/Documents
- Recommendation/Collaborative filtering
- Videos
- Motion Capture
- Speech Perception
- …



Homework

- Problem sets and Computer exercises (PrEx8): choose at least one task among the following 4 optional tasks:
  - (optional 1) Write an essay on Hopfield Network with full mathematical developments.
  - (optional 2) Write an essay on RBM with real-valued and/or count variables with key mathematical developments.
  - (optional 3) Find a package of Hopfield Net and a demo task with real data, and do the computer exercise on the task.
  - (optional 4) Find a package of RBM and a demo task with real data, and do the computer exercise on the task.
- Deadline:
  - Dec. 15 (Wednesday), 23:00

D

单选题 1分 设置

4-min break

A Yes, I’m back to my computer.

B Sorry, not yet.

提交