智能科学

— Chapter 3 —

神经计算

史忠植

中国科学院计算技术研究所
shizz@ics.ict.ac.cn
http://www.intsci.ac.cn/

内容提要

- 引言
- 感知器
- 反传**Back Propagation**
- 递归网络**Recurrent network**
- **Hopfield** 网络
- 自组织映射
- 讨论和展望

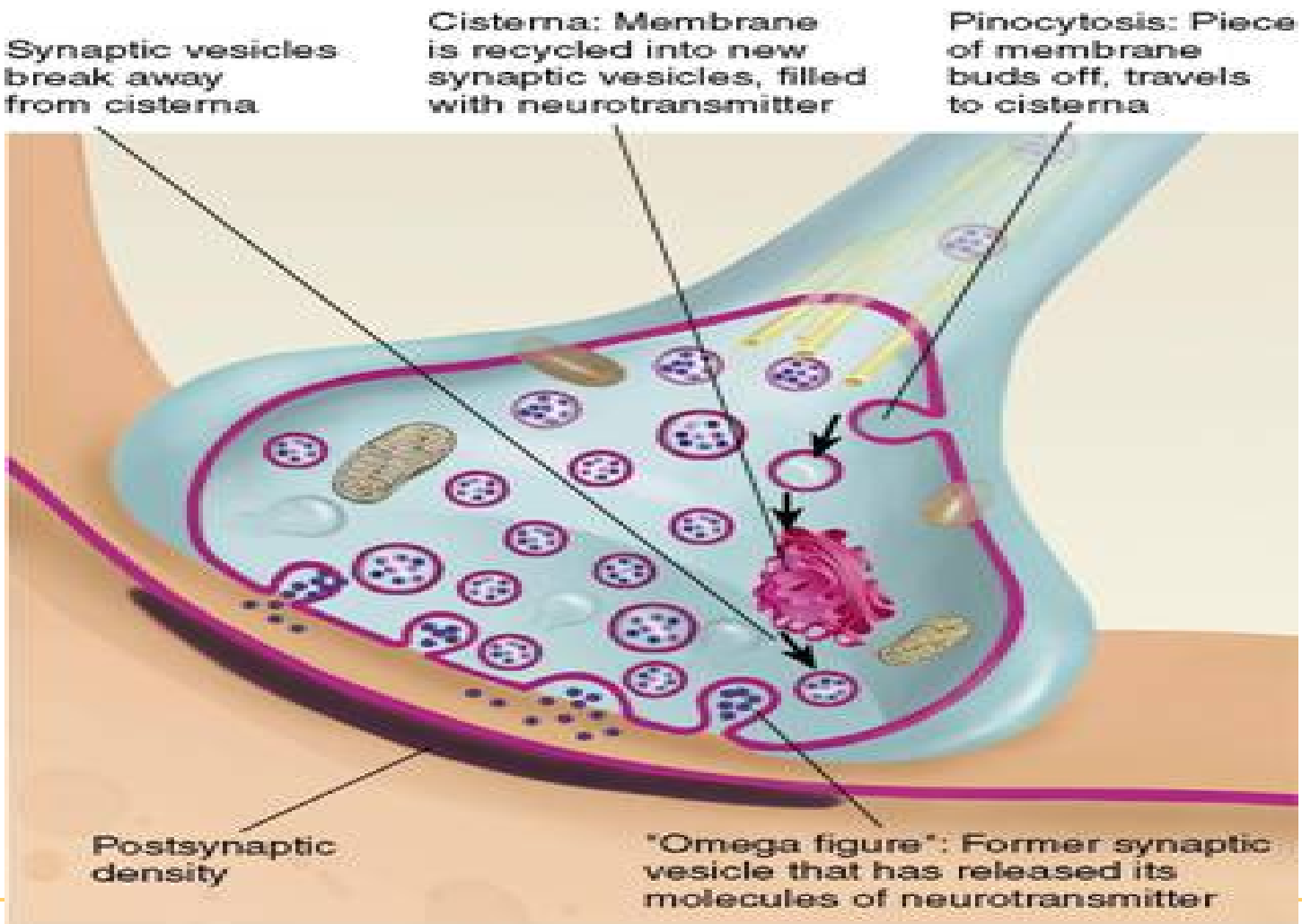
引言

- 一个神经网络是一个由简单处理元构成的规模宏大的并行分布处理器。天然具有存储经验知识和使之可用的特性。

神经网络从两个方面上模拟大脑：

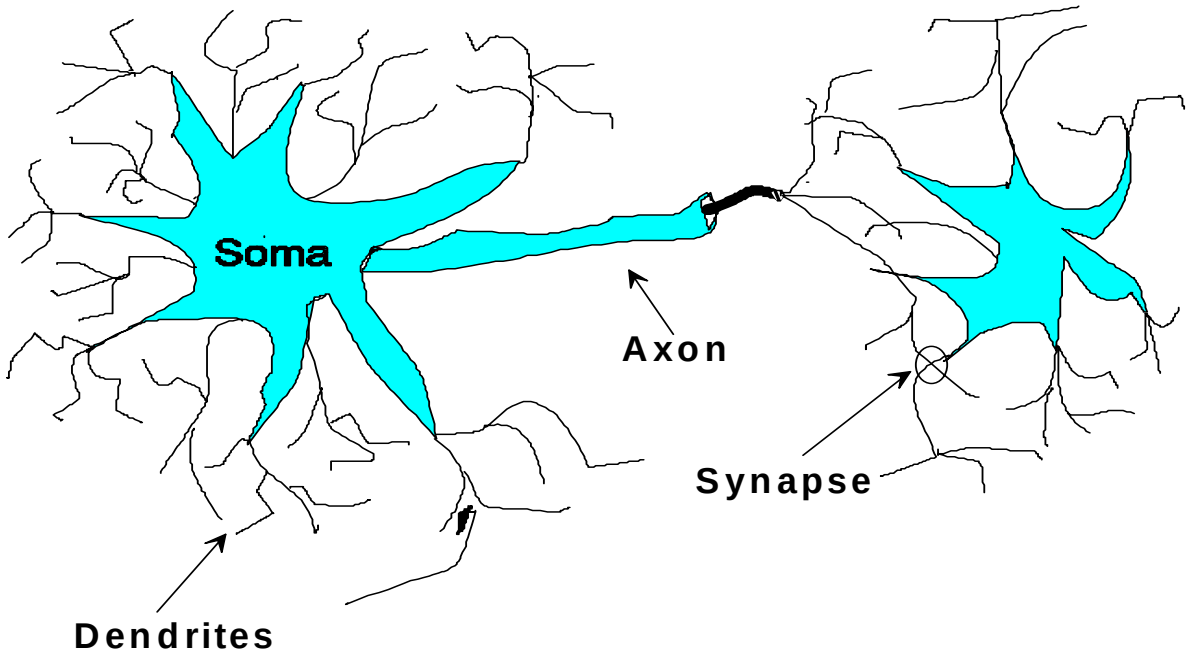
- 神经网络获取的知识是从外界环境中学习得来的。
- 内部神经元的连接强度，即突触权值，用于储存获取的知识。

突触可塑性


Synaptic vesicles break away from cisterna
Cisterna: Membrane is recycled into new synaptic vesicles, filled with neurotransmitter
Pinocytosis: Piece of membrane buds off, travels to cisterna
Postsynaptic density
"Omega figure": Former synaptic vesicle that has released its molecules of neurotransmitter

生物神经系统

- 人脑由大约 100 billion (10^{11}) 神经元组成



人工神经网络

ANN(Artificial Neural Network)是一种基于连接主义机制的人工智能技术。它试图从微观上解决人类认知功能，以探索认知过程的微结构，并在网络层次上模拟人类的思维方式和组织形式。它通过合理的样本训练、学习专家的经验、模拟专家的行为，并通过引入非线性转换函数来求解各种复杂的非线性问题，从而使ANN具有很强的模式识别能力，可克服传统模式识别方法或一般算法在求解问题、处理数据时存在决策界面统计不出或不准确的现象。

人工神经网络

人工神经网络ANN具有学习、联想、自组织、记忆、容错和鲁棒等功能，这不仅可避开建立复杂的数学模型和进行繁琐的数学推理，而且对信息不完全的数据（资料）进行ANN模型训练和处理，较之采用常规方法往往能获得良好的结果。ANN作为模拟人的智能和形象思维能力的一种重要途径和方法，在模式识别、信号处理、自动控制等领域获得应用取得显著成效，并在非线性建模和石油勘探开发等复杂问题求解方面有着广阔的应用前景。

神经计算

（1）可以充分逼近任意复杂的非线性关系

（2）所有定量或定性的信息都等势分布贮存于网络内的各神经元，故有很强的鲁棒性和容错性

（3）采用并行分布处理方法，使得快速进行大量运算成为可能

（4）可学习和自适应不知道或不确定的系统

（5）能够同时处理定量、定性知识。

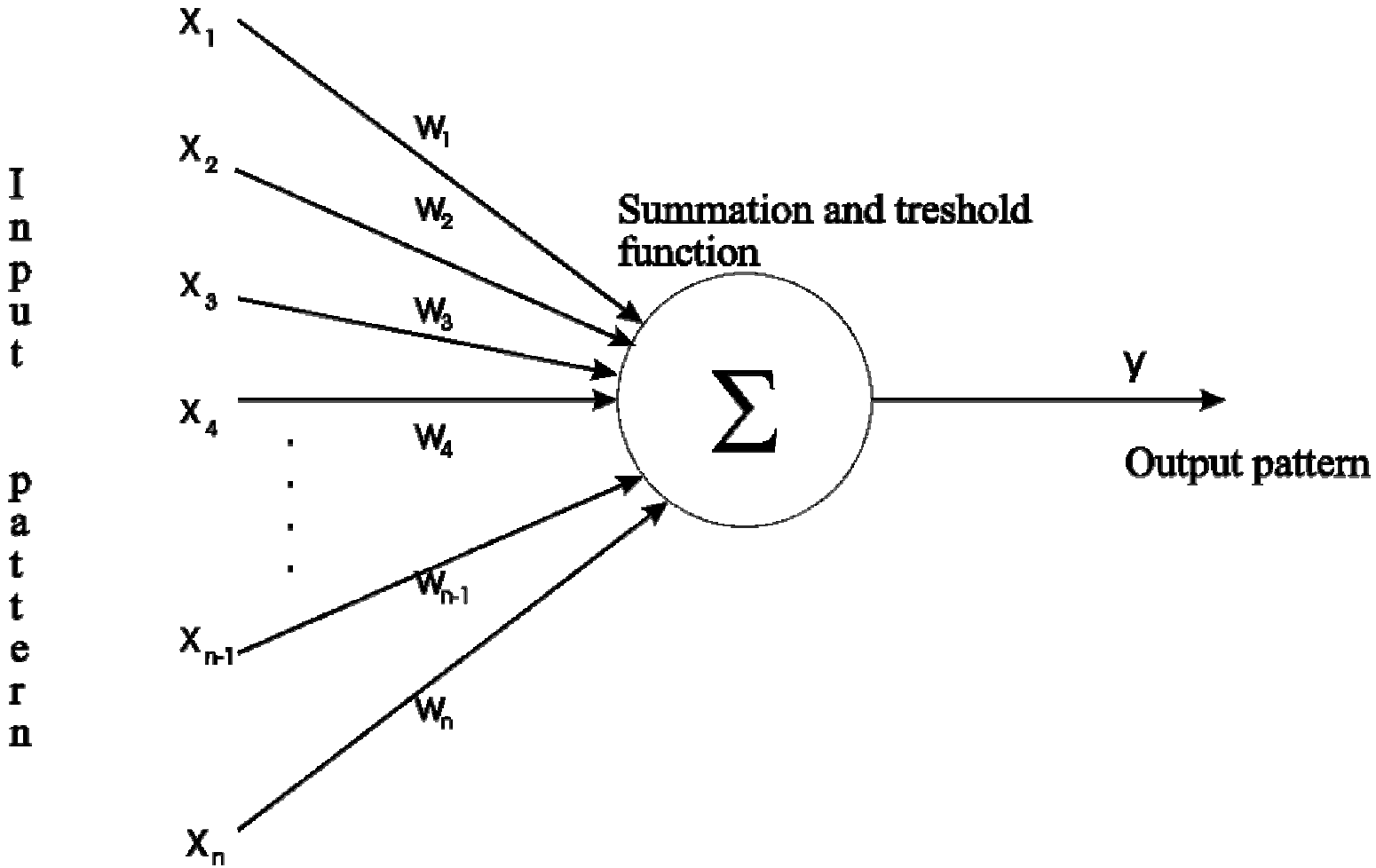
神经网络的维数

- Various types of neurons
- Various network architectures
- Various learning algorithms
- Various applications

神经计算

- 40年代心理学家Mcculloch和数学家Pitts合作提出的兴奋与抑制型神经元模型和Hebb提出的神经元连接强度的修改规则，他们的研究结果至今仍是许多神经网络模型研究的基础。

McCulloch-Pitts Neuron



神经计算

- 50年代、60年代的代表性工作是Rosenblatt的感知机和Widrow的自适应性元件Adaline。
- 1969年，Minsky和Papert合作发表了颇有影响的Perceptron一书，

  Failure with linearly separable problems

  – XOR [(X1=T & X2=F) or (X1=F & X2=T)]

  – Weakness repaired with hidden layers

  (Minsky, M. and Papert, S., *Perceptrons*, MIT Press, Cambridge, 1969).

- 得出了消极悲观的论点，加上数字计算机正处于全盛时期并在人工智能领域取得显著成就，70年代人工神经网络的研究处于低潮。

神经计算

- 多层网络BP算法
- Hopfield网络模型
- 自适应共振理论（ART）
- 自组织特征映射理论
- Hinton 等人最近提出了 Helmboltz 机
- 徐雷提出的 Ying-Yang 机理论模型
- 甘利俊一（ S.Amari） 开创和发展的基于统计流形的方法应用于人工神经网络的研究,

神经计算

Late 1980's - NN re-emerge with Rumelhart and McClelland (Rumelhart, D., McClelland, J., **Parallel and Distributed Processing**, MIT Press, Cambridge, 1988.

神经元建模

神经元是神经网络中基本的信息处理单元，他由下列部分组成:

1 一组突触和联结，联结具有权值

$W_1, W_2, \ldots, W_m$

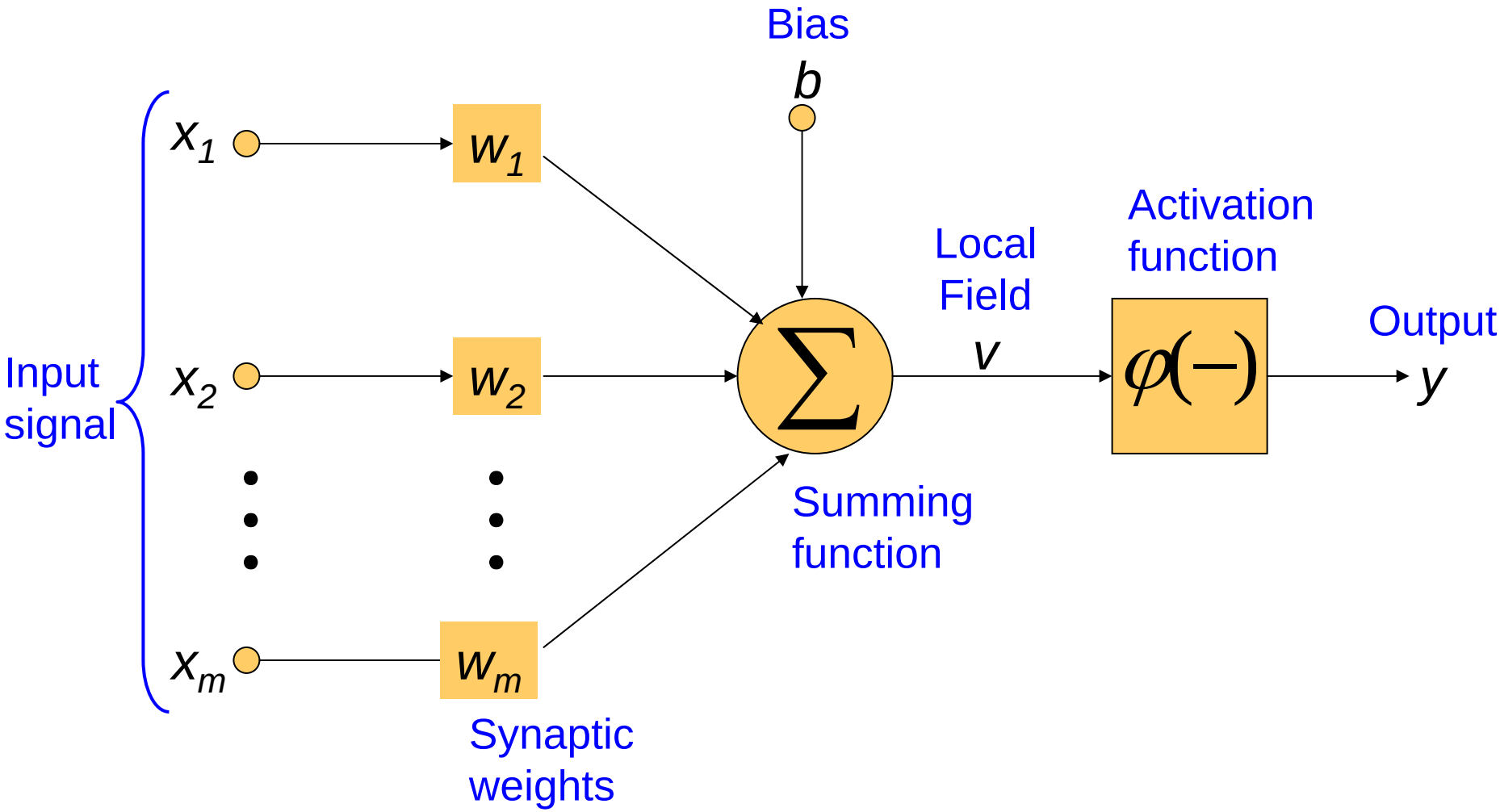
2 通过加法器功能，将计算输入的权值之和

$$u = \sum_{j=1}^{m} w_j x_j$$

3 激励函数限制神经元输出的幅度

$$y = \varphi(u + b)$$

神经元

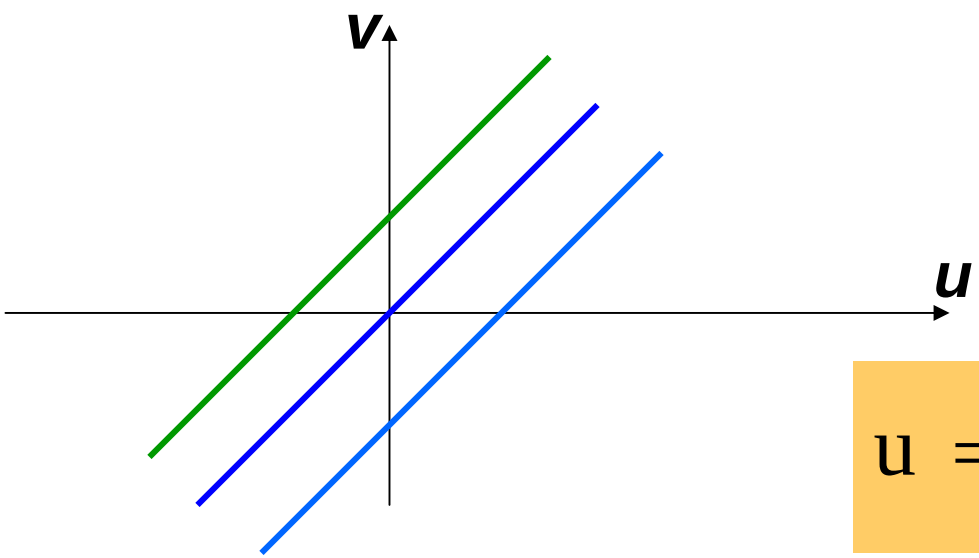

Bias
b
x_1
w_1
Input signal
x_2
w_2
x_m
w_m
Synaptic weights
Summing function
Local Field
v
Activation function
$\varphi(-)$
Output
y

神经元偏置

- Bias ***b*** has the effect of applying an affine transformation to ***u***

$$v = u + b$$

- ***v*** is the **induced field** of the neuron



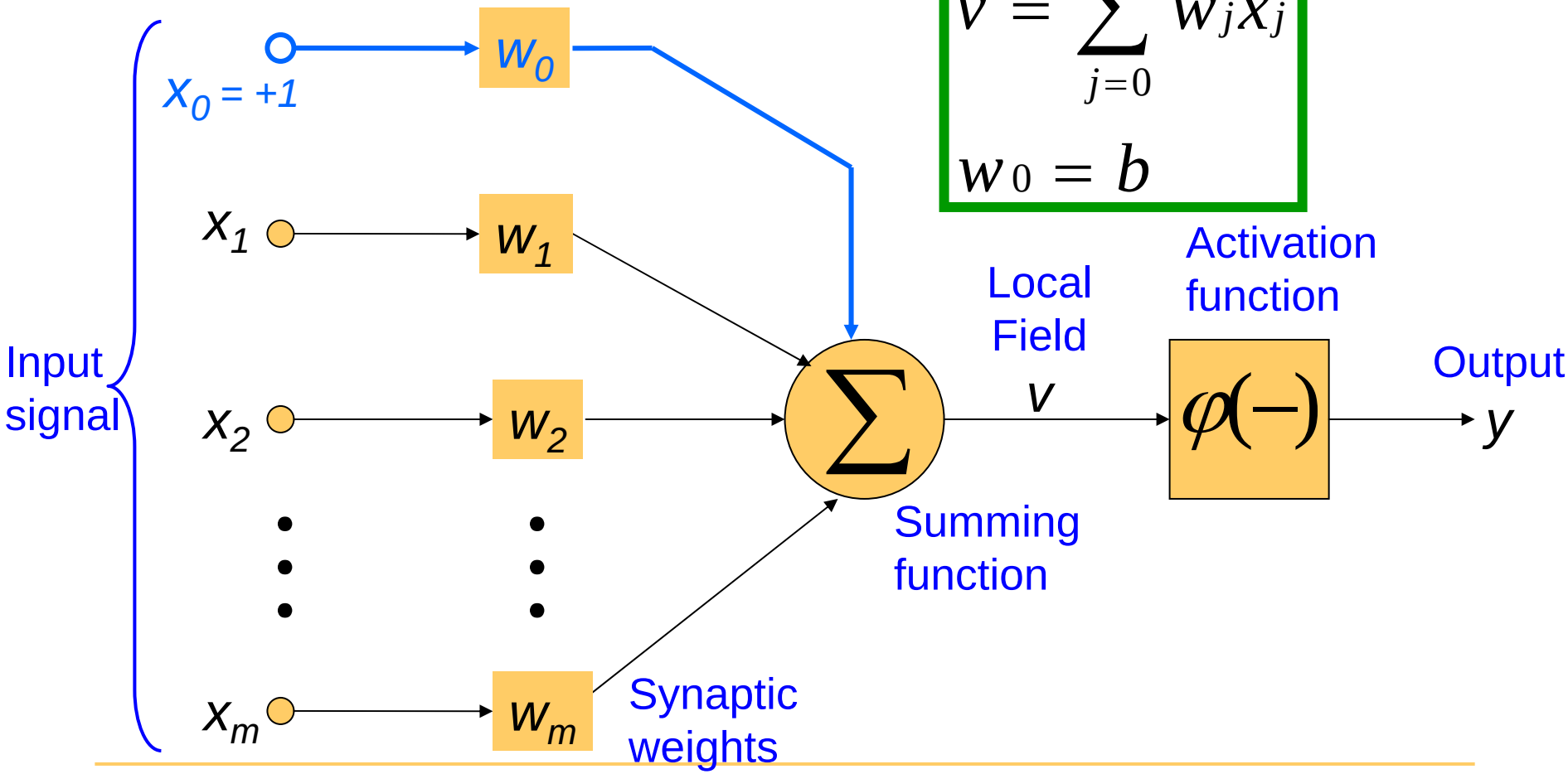
$$u = \sum_{j=1}^{m} w_j x_j$$

偏置作为额外输入

- Bias is an external parameter of the neuron. Can be modeled by adding an extra input.

$$v = \sum_{j=0}^{m} w_j x_j$$

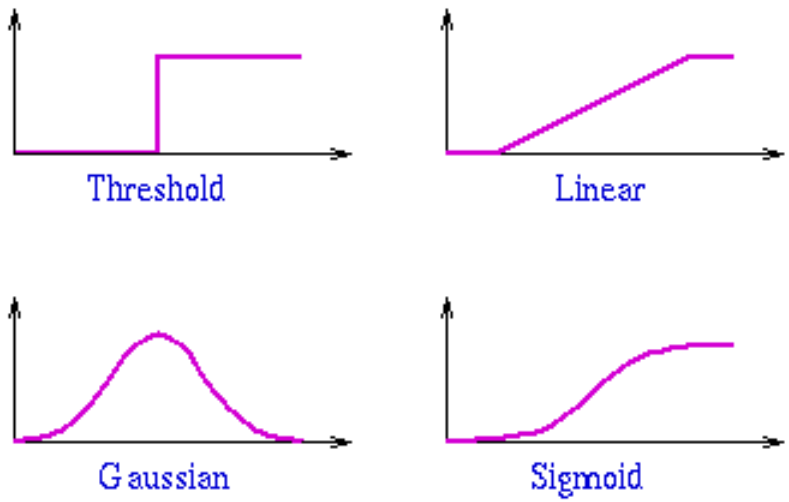
$$w_0 = b$$



Hebb学习假说

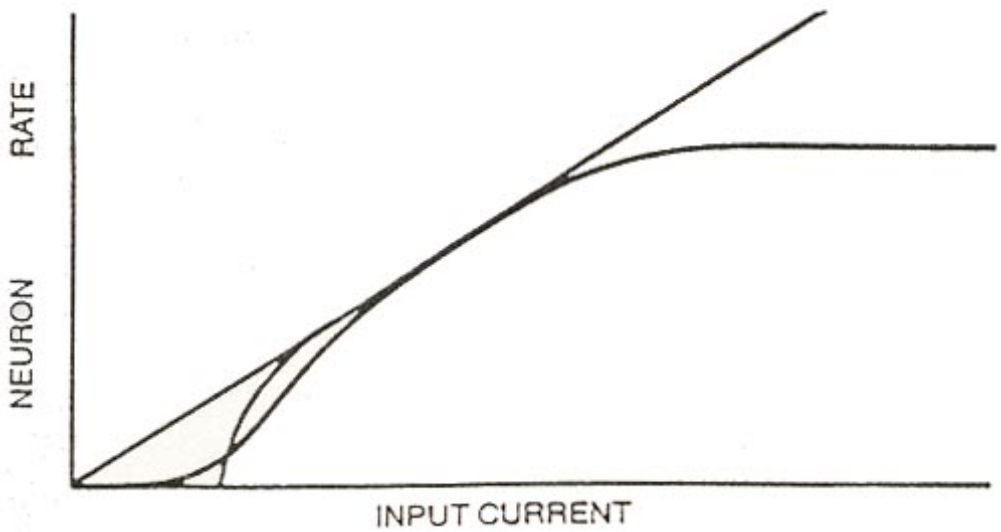
- 1949年，Hebb的书《行为组织学》，他在里面第一次清楚说明了突触修正的生理学习规则。特别的， Hebb提出大脑的连接随着生物学会不同功能任务而连续地变化的， 神经组织就是由这种变化创建起来的. Hebb 继承了 Ramony 和 Cajal 早期的假设并引入自己的现在著名的学习假说，两个神经元之间的可变突触被突触两端神经元的重复的激活加强了。Hebb的书在心理学家中有广泛的影响，但遗憾的是对工程界却影响很少或没有。

一般规则



- Simplify the complexity
- Task specific

生物点火速率



- Average Firing Rate depends on biological effects such as leakage, saturation, and noise

训练方法

- Supervised training
- Unsupervised training
  - Self-organization

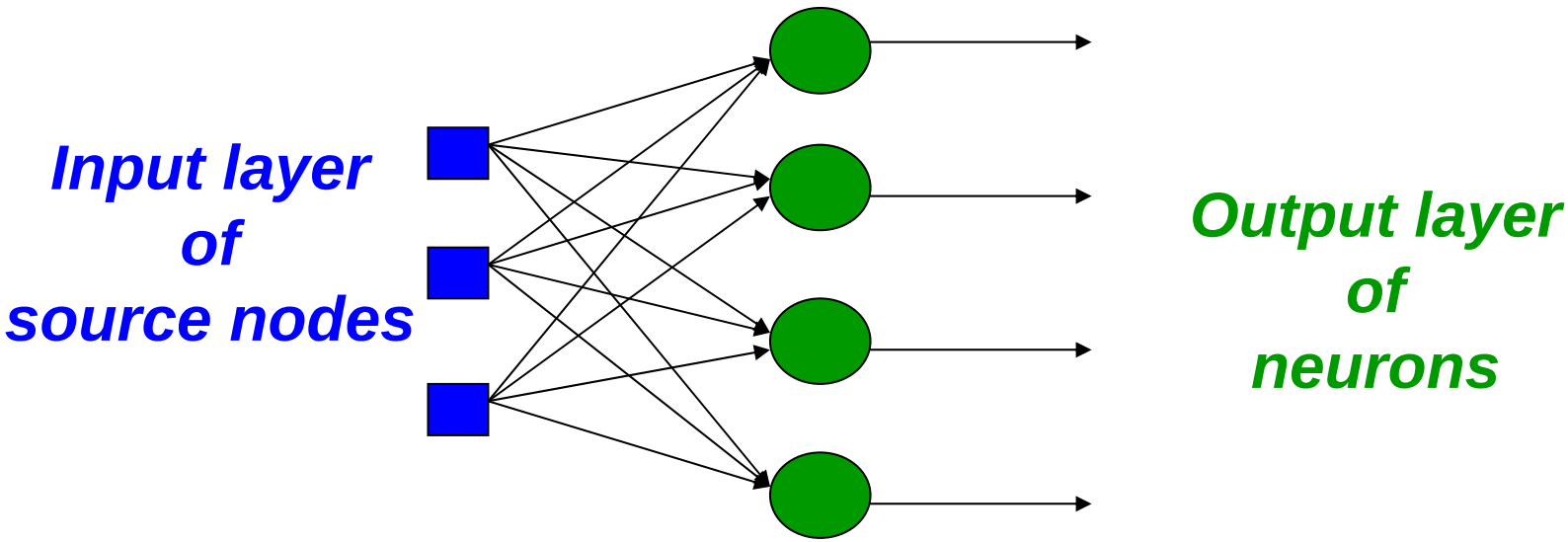
- Back-Propagation
- Simulated annealing
- Credit assignment

内容提要

- 引言
- 感知器
- 反传**Back Propagation**
- 递归网络**Recurrent network**
- **Hopfield** 网络
- 自组织映射
- 讨论和展望

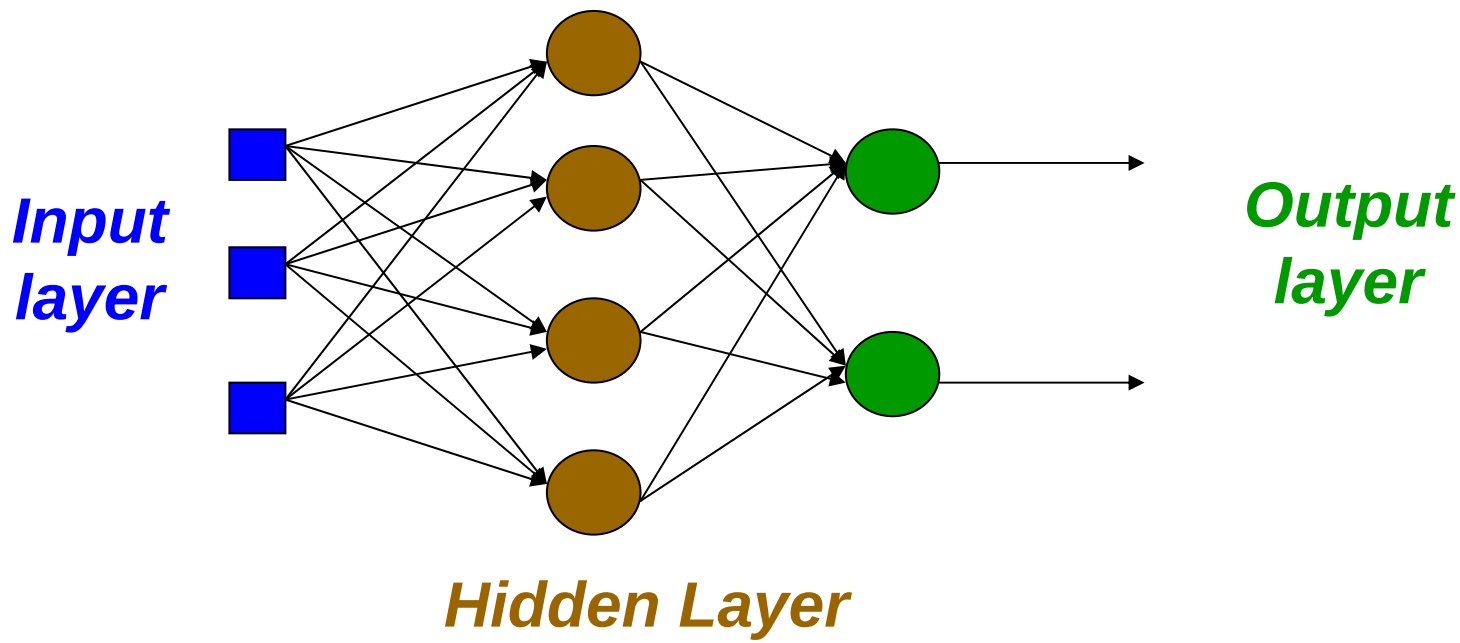
感知器：单层前馈网络

Rosenblatt's Perceptron: a network of processing elements (PE):



感知器：单层前馈网络

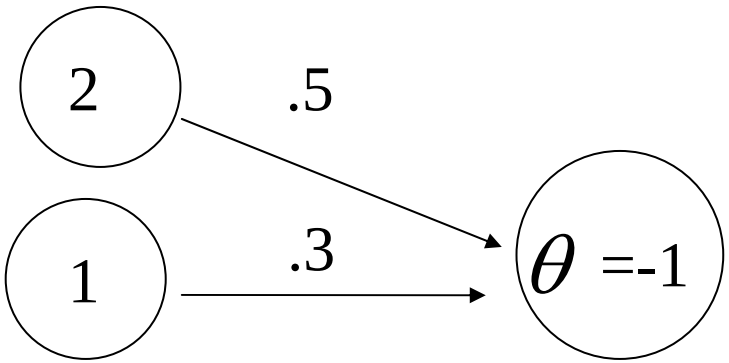
3-4-2 Network



感知器：学习规则

- $Err = T - O$
  - O is the predicted output
  - T is the correct output
- $W_j \leftarrow W_j + \alpha * I_j * Err$
  - I_j is the activation of a unit j in the input layer
  - α is a constant called the learning rate

感知器



$$2(0.5) + 1(0.3) + -1 = 0.3 \text{ , } O=1$$

Learning Procedure:

Randomly assign weights (between 0-1)

Present inputs from training data

Get output O, nudge weights to gives results toward our desired output T

Repeat; stop when no errors, or enough epochs completed

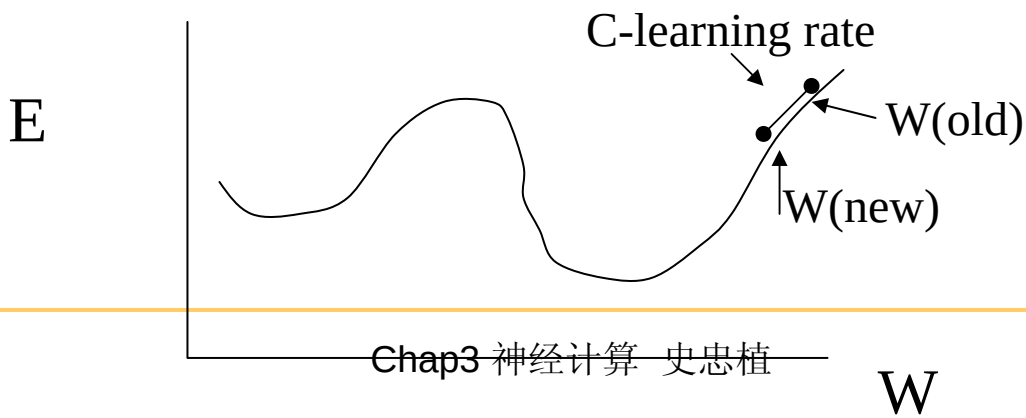
最小均方学习

LMS = Least Mean Square learning Systems, more general than the previous perceptron learning rule. The concept is to minimize the total error, as measured over all training examples, P. O is the raw output, as calculated by $\sum_i w_i I_i + \theta$

$$Dis\tan ce(LMS) = \frac{1}{2}\sum_P (T_P - O_P)^2$$

E.g. if we have two patterns and
T1=1, O1=0.8, T2=0, O2=0.5 then D=(0.5)[(1-0.8)^2+(0-0.5)^2]=.145

We want to minimize the LMS:



作用函数

- To apply the LMS learning rule, also known as the delta rule, we need a differentiable activation function.

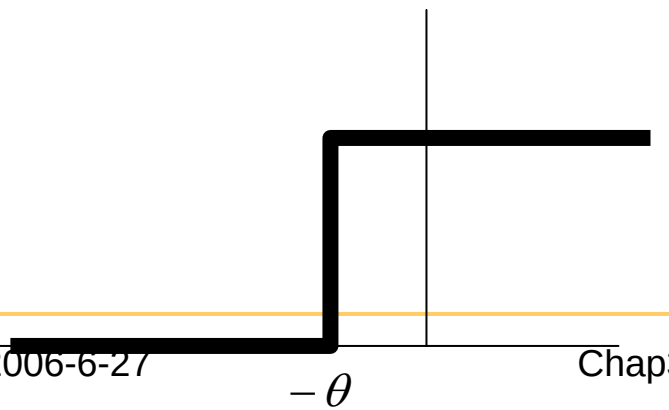
$$\Delta w_k = cI_k\left(T_j - O_j\right)f'\left(ActivationFunction\right)$$

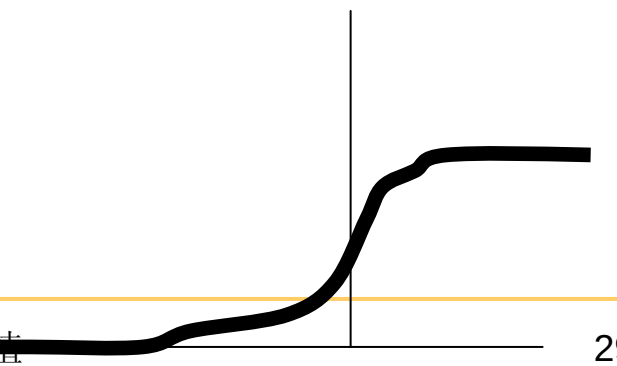
Old:

$$O = \begin{cases} 1 : \sum_i w_i I_i + \theta > 0 \\ 0 : otherwise \end{cases}$$

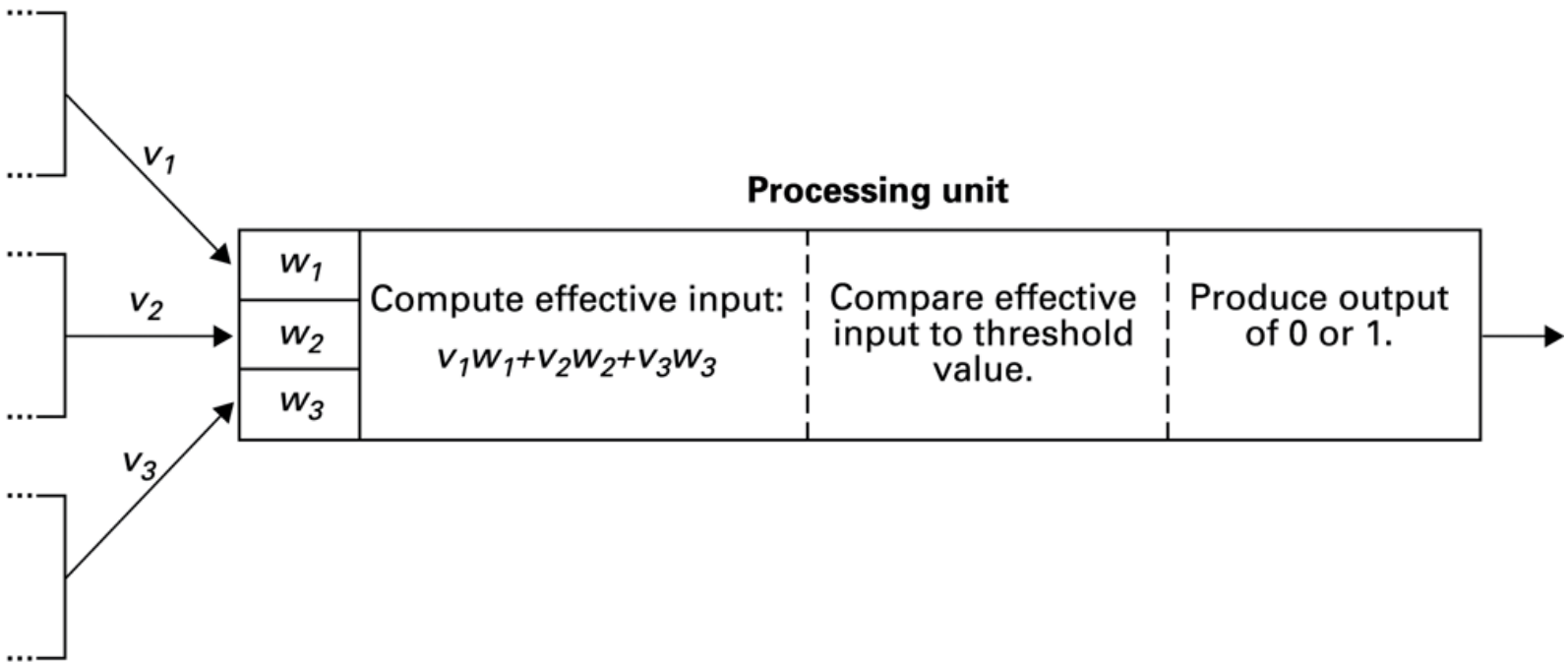
New:

$$O = \frac{1}{1 + e^{-\sum_i w_i I_i + \Theta}}$$

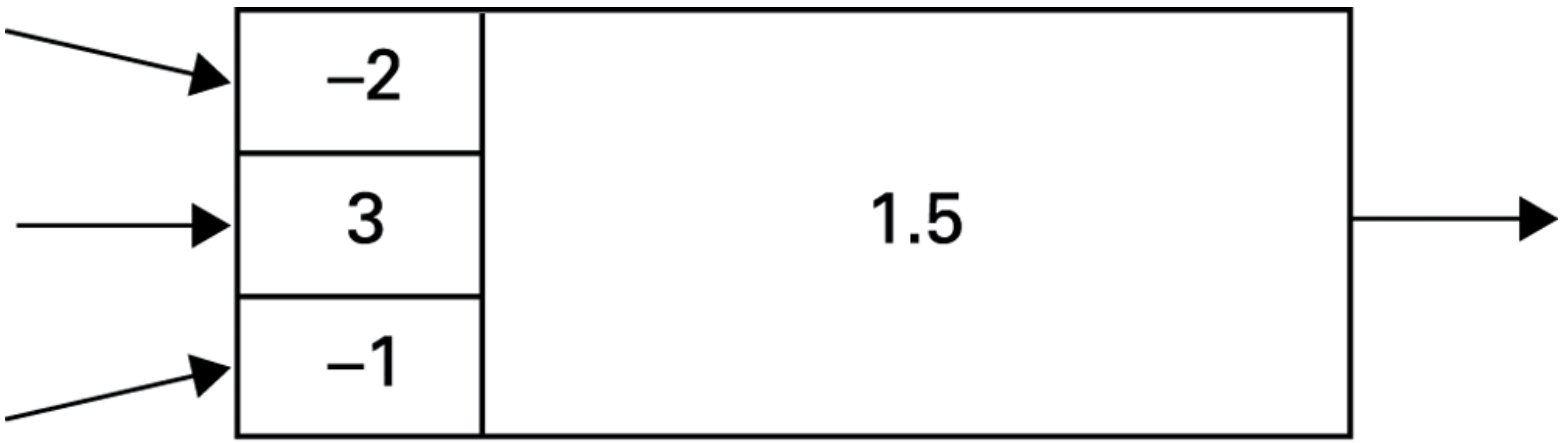




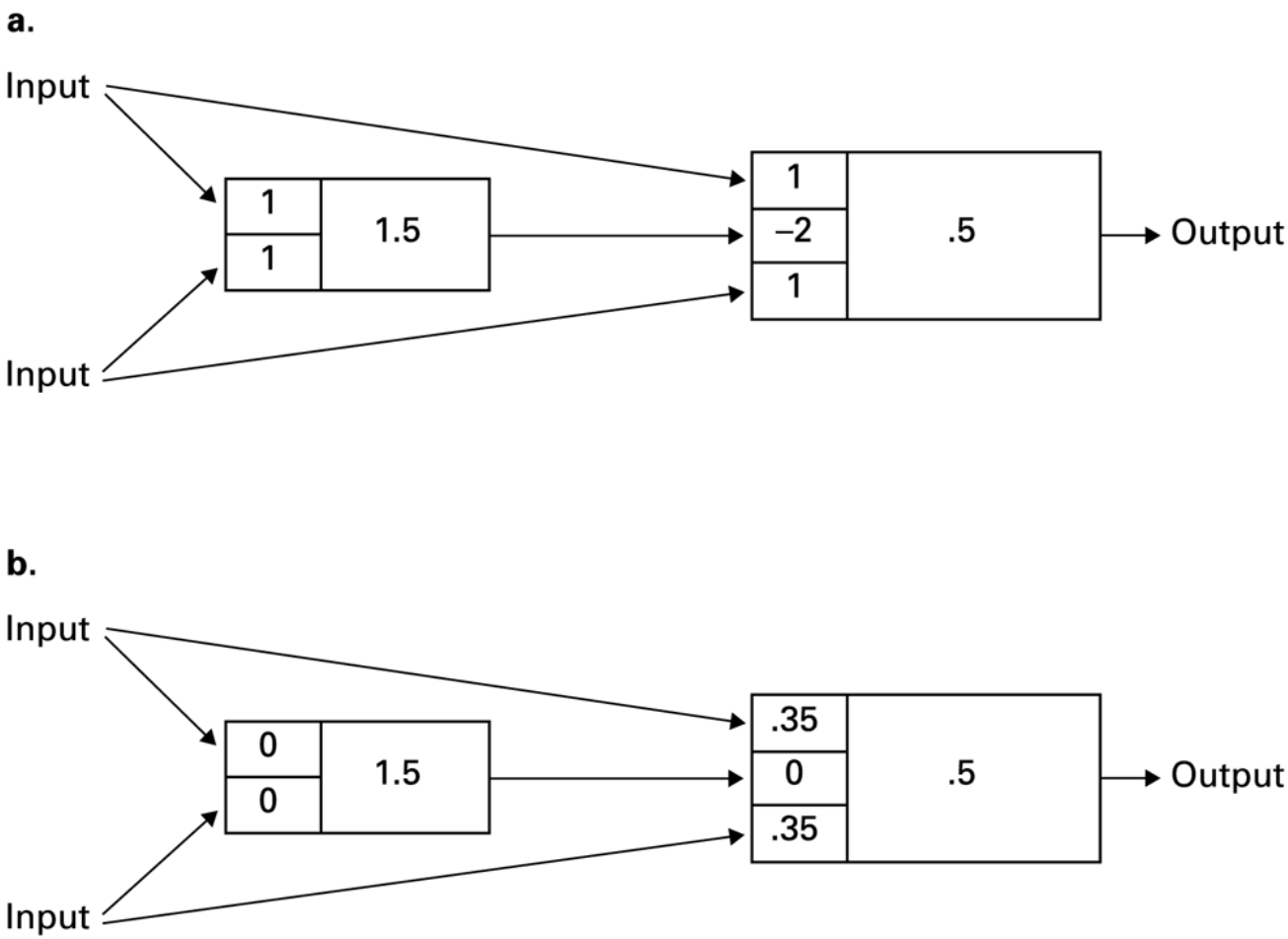
作用处理单元


Processing unit
v_1
v_2
v_3
w_1
w_2
w_3
Compute effective input:
$v_1w_1+v_2w_2+v_3w_3$
Compare effective input to threshold value.
Produce output of 0 or 1.

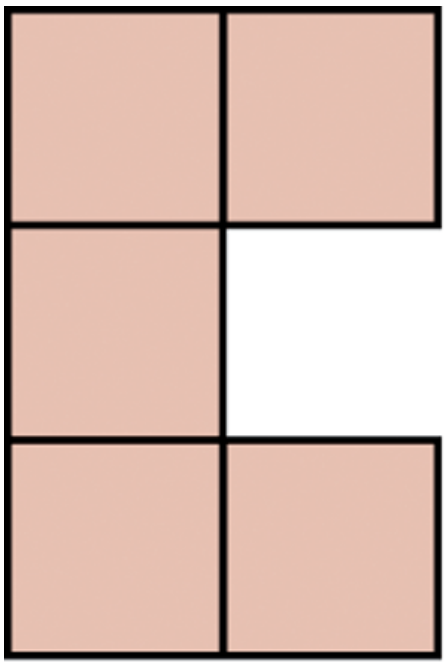
处理单元的表达

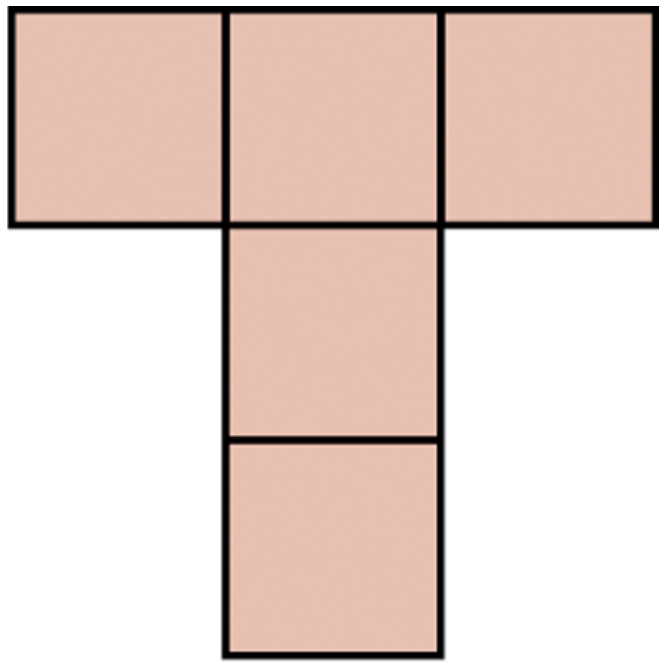


神经网路结构


a.
Input
1
1
1.5
1
–2
1
.5
Output
Input
b.
Input
0
0
1.5
.35
0
.35
.5
Output
Input

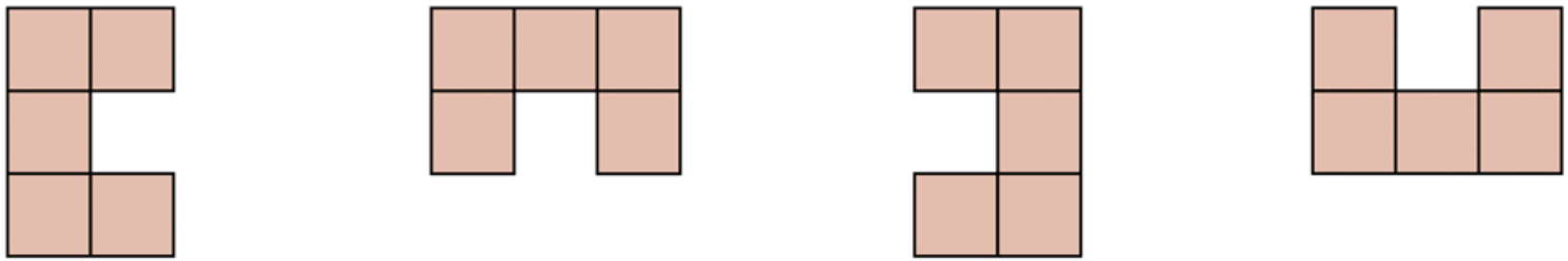
大写 C 和 T



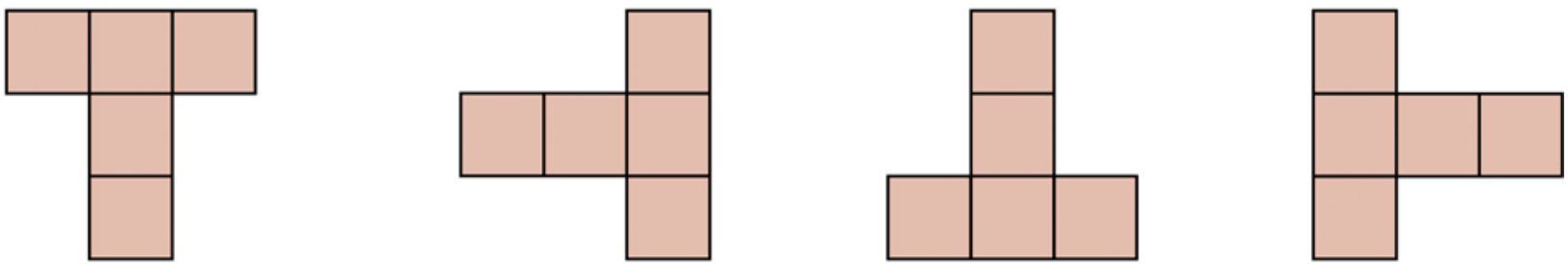


不同取向字母C和T

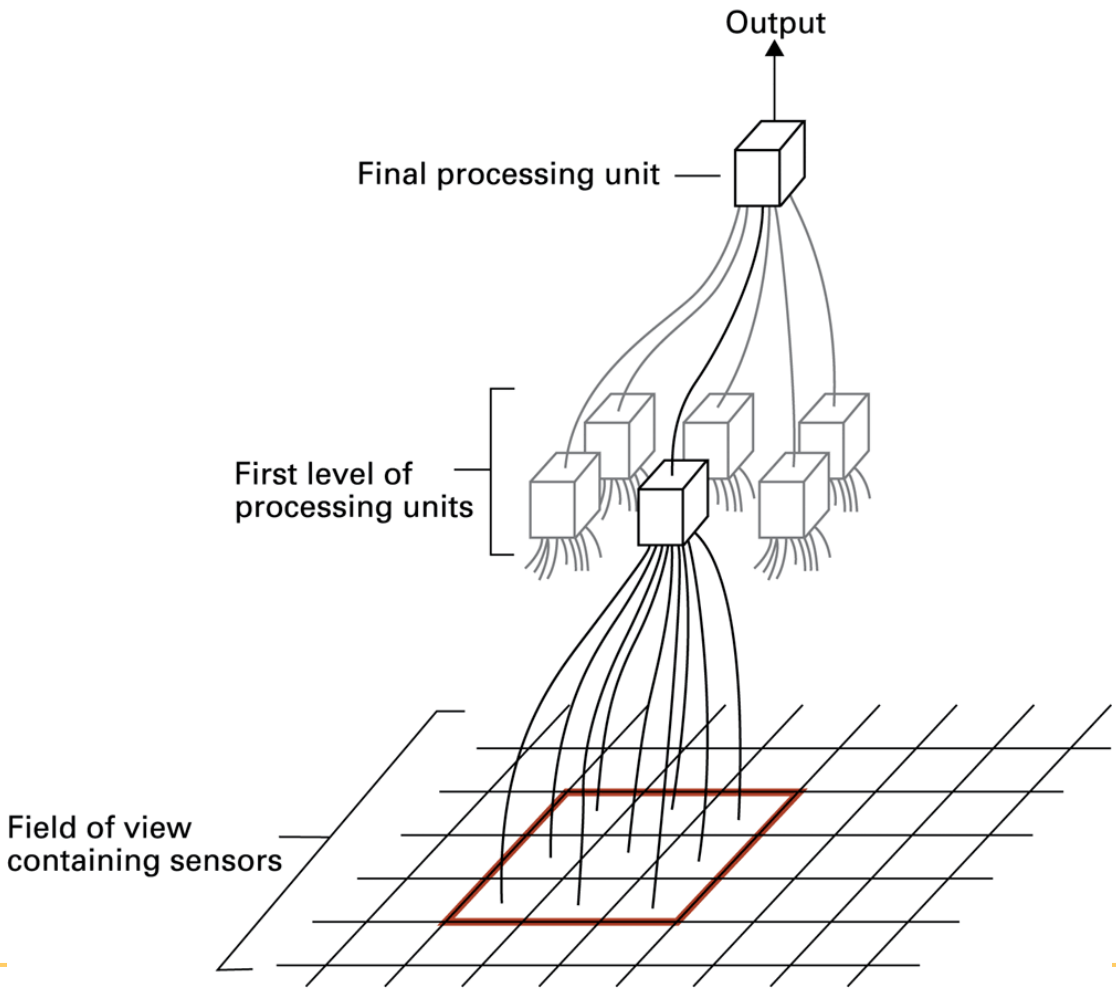
a.



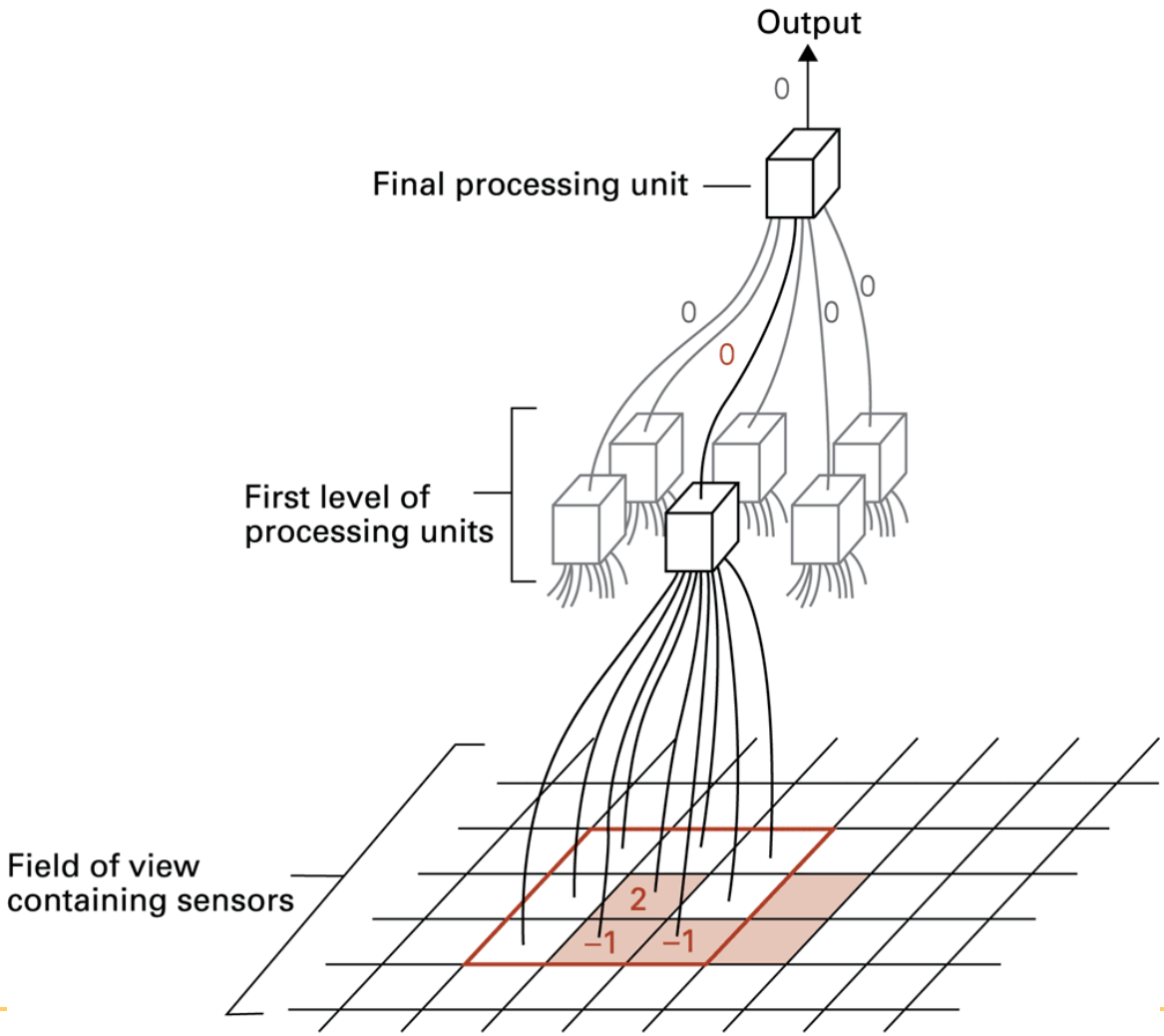
b.



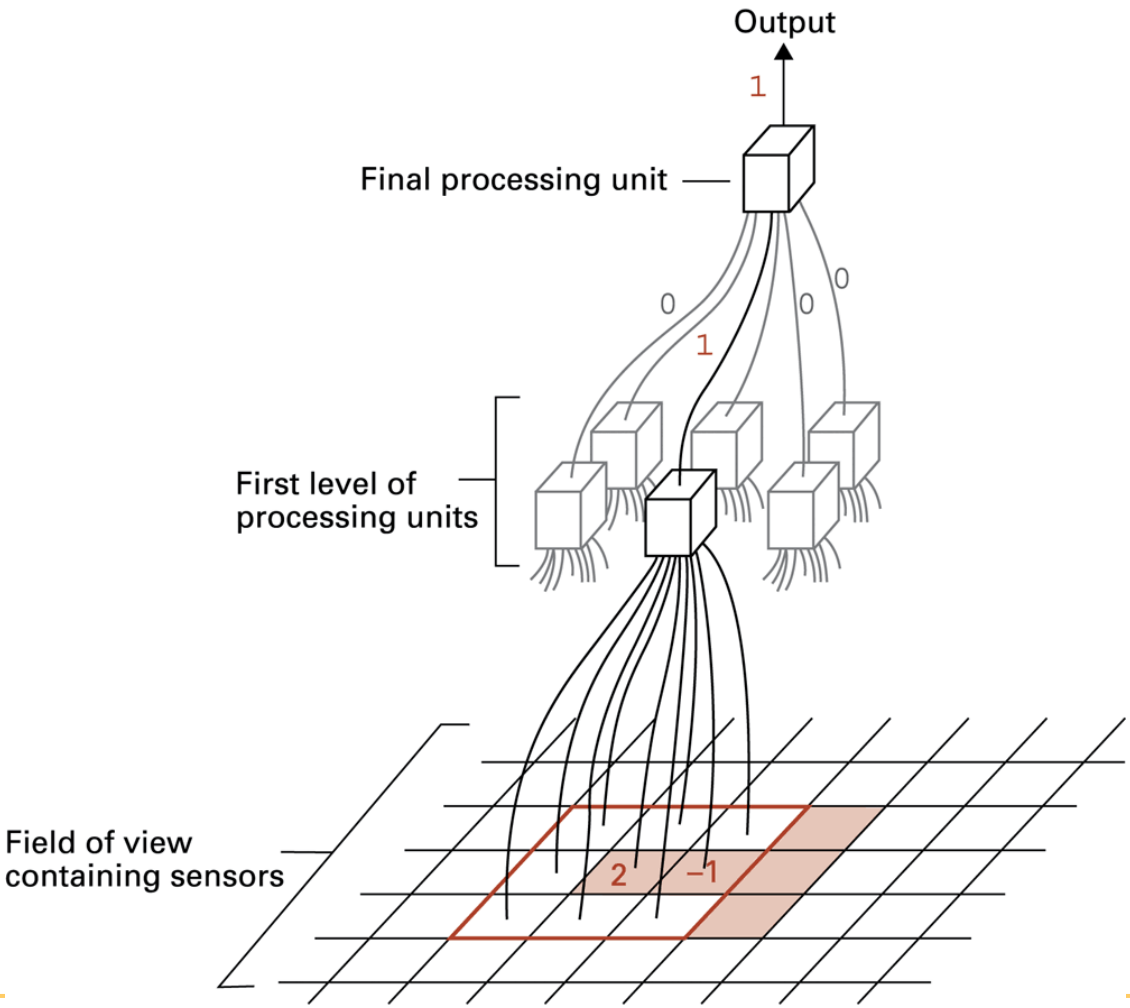
字符识别系统


Output
Final processing unit
First level of
processing units
Field of view
containing sensors

观察域中的字母C


Output
0
Final processing unit
0
0
0
0
First level of
processing units
Field of view
containing sensors
2
–1
–1

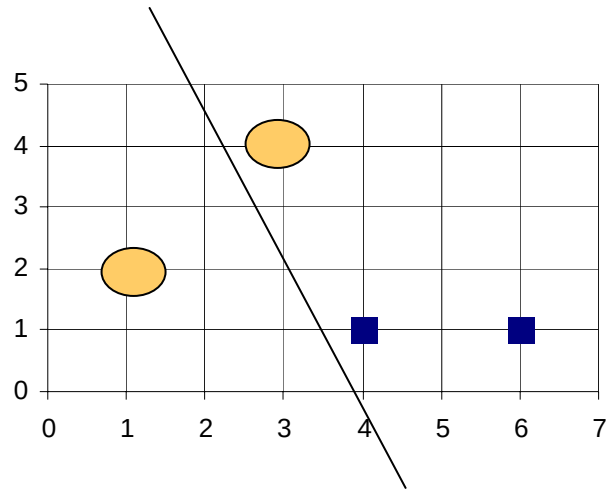
观察域中的字母 T


Output
1
Final processing unit
0
1
0
0
First level of
processing units
Field of view
containing sensors
2
–1

感知器分类器

- For example, suppose there are 4 training data points (with 2 positive examples of the class and 2 negative examples)

X1	X2	Class
3	4	0
6	1	1
4	1	1
1	2	0

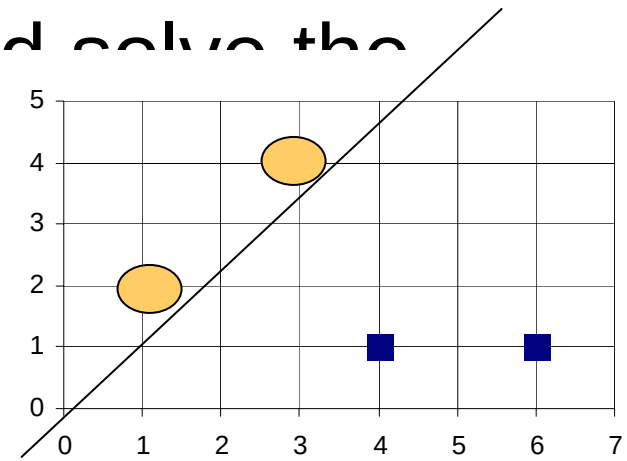


- The initial random value of the weights will probably not divide these points accurately

感知器分类器

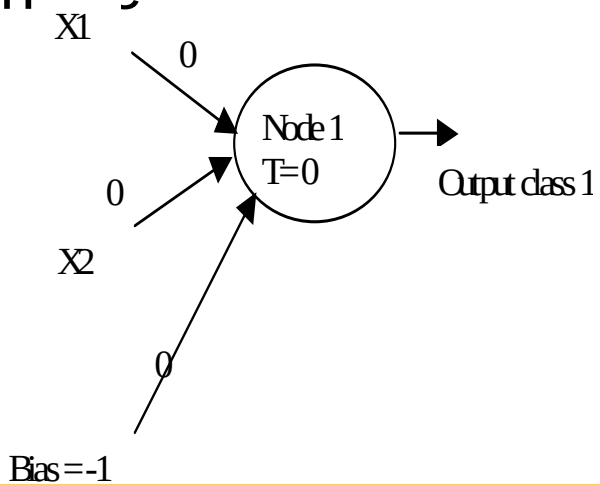
- But during training the weight values are changed, based on the reduction of ‘error’
- Eventually a line can be found that does divide the points and solve the classification task

eg 4X1 + -3.5X2 = 0

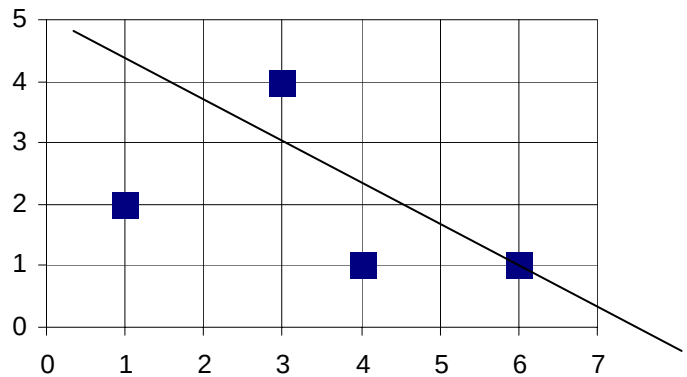


二分感知器

- Dichotomising Perceptrons divide data into 2 groups.
- They have a single output node and a binary response for that node



- With 2 input variables the decision line is placed in a 2-dimensional space



- The presence of a bias input means the line does not have to pass through the origin

二分感知器

- The set of weights can be represented as a matrix, W
  - normally each row corresponds to a weight vector leading to an output node
  - With a single output node there would be a single row in the weight matrix
- The presentation of a set of inputs, X, can be represented as a matrix multiplication W.X
  - Normaly each column of the matrix corresponds to a set of inputs feeding into the network
  - This means a 1*n matrix is multiplied by a n*1 matrix producing the single weighted sum that is the node activation

二分感知器

- For example, a set of weights W = (1, 2, -1)
- The input values are X1=2, X2 = 0 and the bias constant is –1
- So the multiplicative sum the node receives is W.X

$$\begin{pmatrix} 1 & 2 & -1 \end{pmatrix} \begin{pmatrix} 2 \\ 0 \\ -1 \end{pmatrix} = \begin{pmatrix} 3 \end{pmatrix}$$

- f(W.X) represents the application of the transfer function to the result of the matrix multiplication. ie f(3) and typically only makes sense for a single value

二分感知器训练

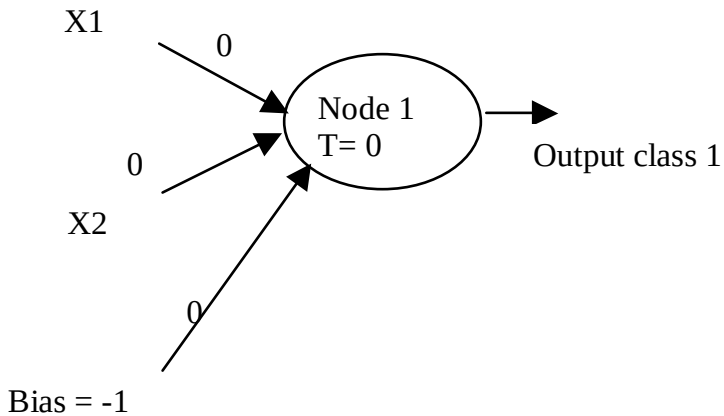
1. Initialise weights to zero
2. Propagate the input and compute actual output for the node (o)
3. Compare actual output (o) with desired output (d)
4. Change weights if desired value not obtained using

$$\mathbf{wt+1 = wt + \eta *(d-o)* xt}$$

   – Large η value gives large weight change (and potentially quicker training);Small value are potentially slower. η may be large at start of training and small at the end
   – Also big errors (d – o) cause big weight changes
5. Repeat from 2 until all inputs presented and weights no longer change

训练实例

- Single piece of training data. Input X = (1.5, -1) with bias input X is (1.5, -1, -1). Target output is 0.
- Initial weight vector $W_{t=0}$ = (0, 0, 0) and η = 1, T is transfer function, T = 0



- f(W*X) is f(0*1.5 + 0*-1 + 0*-1) = f(0) = 1.
- Incorrect so change weights

$W_{t=1} = (0, 0, 0) + 1*(0 - 1) * (1.5, -1, -1) = (-1.5, 1, 1)$

训练过程

- Present next input, say X = (1.5, -1, -1) again with target output 0.
- Compute f(W*X) with the new weight vector

$$W_{t=1} = (-1.5, 1, 1)$$

- This is f(-1.5*1.5 + 1*-1 + 1*-1) = f(-2.25+-1+-1)
  = f(-4.25) =0.
- There is now no error with the target value and so no further weight change is needed; training is complete

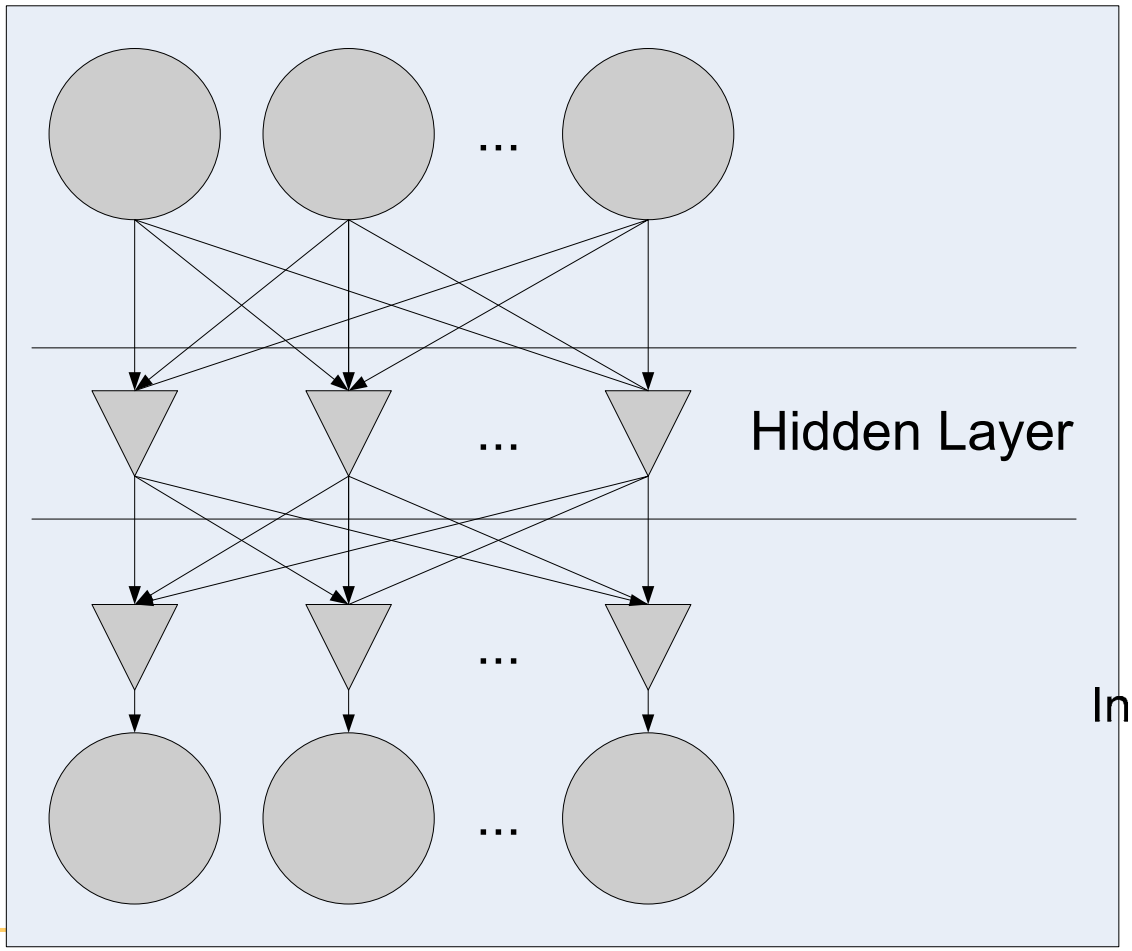
(note how the decision line / weight matrix has evolved as a search was made for the correct weights)

内容提要

- 引言
- 感知器
- 反传**Back Propagation**
- 递归网络**Recurrent network**
- **Hopfield** 网络
- 自组织映射
- 讨论和展望

反传网络 (BP)

- Inputs are put through a ‘Hidden Layer’ before the output layer
- All nodes connected between layers



学习规则

- Measure error
- Reduce that error
  - By appropriately adjusting each of the weights in the network

BP 网络

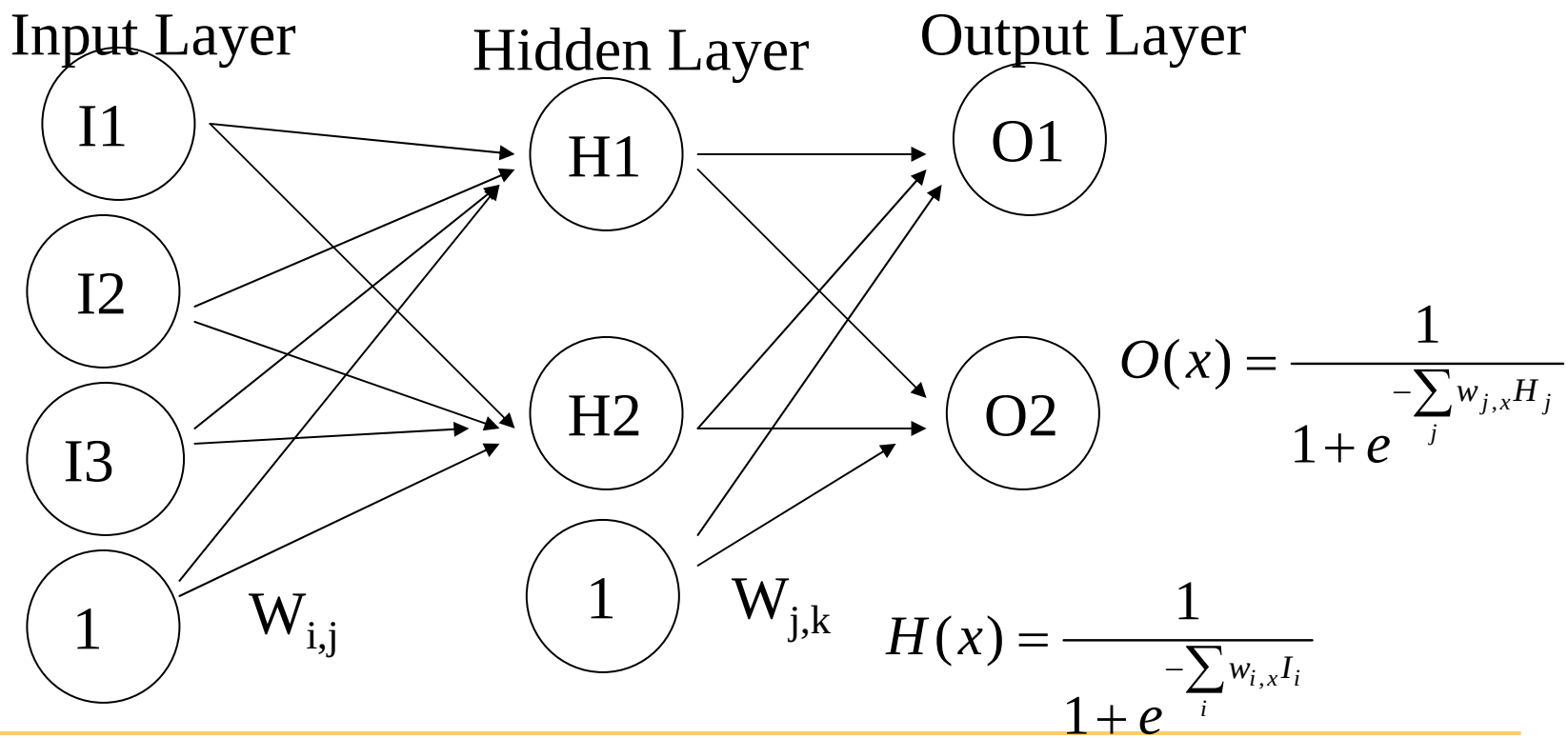
- Forward Pass:
  - Error is calculated from outputs
  - Used to update output weights
- Backward Pass:
  - Error at hidden nodes is calculated by back propagating the error at the outputs through the new weights
  - Hidden weights updated

学习规则

- $Err_i = T_i - O_i$
- $W_{j,i} \leftarrow W_{j,i} + \alpha * a_j * \Delta_i$
  - $\Delta_i = Err_i * g'(in_i)$
  - g' is the derivative of the activation function g
  - a_j is the activation of the hidden unit
- $W_{k,j} \leftarrow W_{k,j} + \alpha * I_k * \Delta_j$
  - $\Delta_j = g'(in_j) * \Sigma_i W_{j,i} * \Delta_i$

BP网路

- To bypass the linear classification problem, we can construct *multilayer* networks. Typically we have *fully connected*, *feedforward* networks.



BP 网络

We had computed:

$$\Delta w_k = cI_k(T_j - O_j)f'(ActivationFunction); \quad f = \left(\frac{1}{1+e^{-sum}}\right)$$

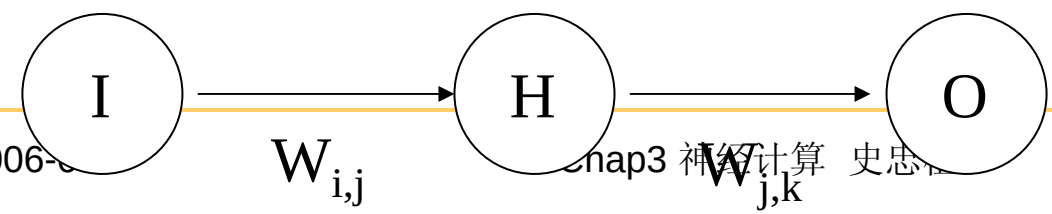
$$\Delta w_k = cI_k(T_j - O_j)(f(sum)(1 - f(sum)))$$

For the Output unit k, f(sum)=O(k). For the output units, this is:

$$\Delta w_{j,k} = cH_j(T_k - O_k)O_k(1 - O_k)$$

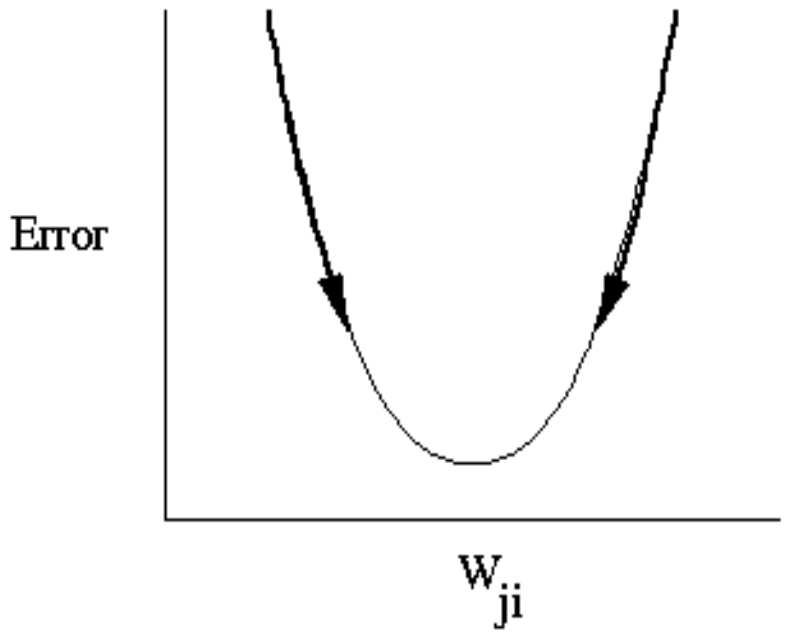
For the Hidden units (skipping some math), this is:

$$\Delta w_{i,j} = cH_j(1 - H_j)I_i\sum_k (T_k - O_k)O_k(1 - O_k)w_{j,k}$$



BP网络学习规则

- $E = 1/2 \sum_i (T_i - O_i)^2$
- $$\frac{\partial E}{\partial W_{k,j}} = - I_k * \Delta_j$$



BP 算法

Learning Procedure:

1. Randomly assign weights (between 0-1)

2. Present inputs from training data, propagate to outputs

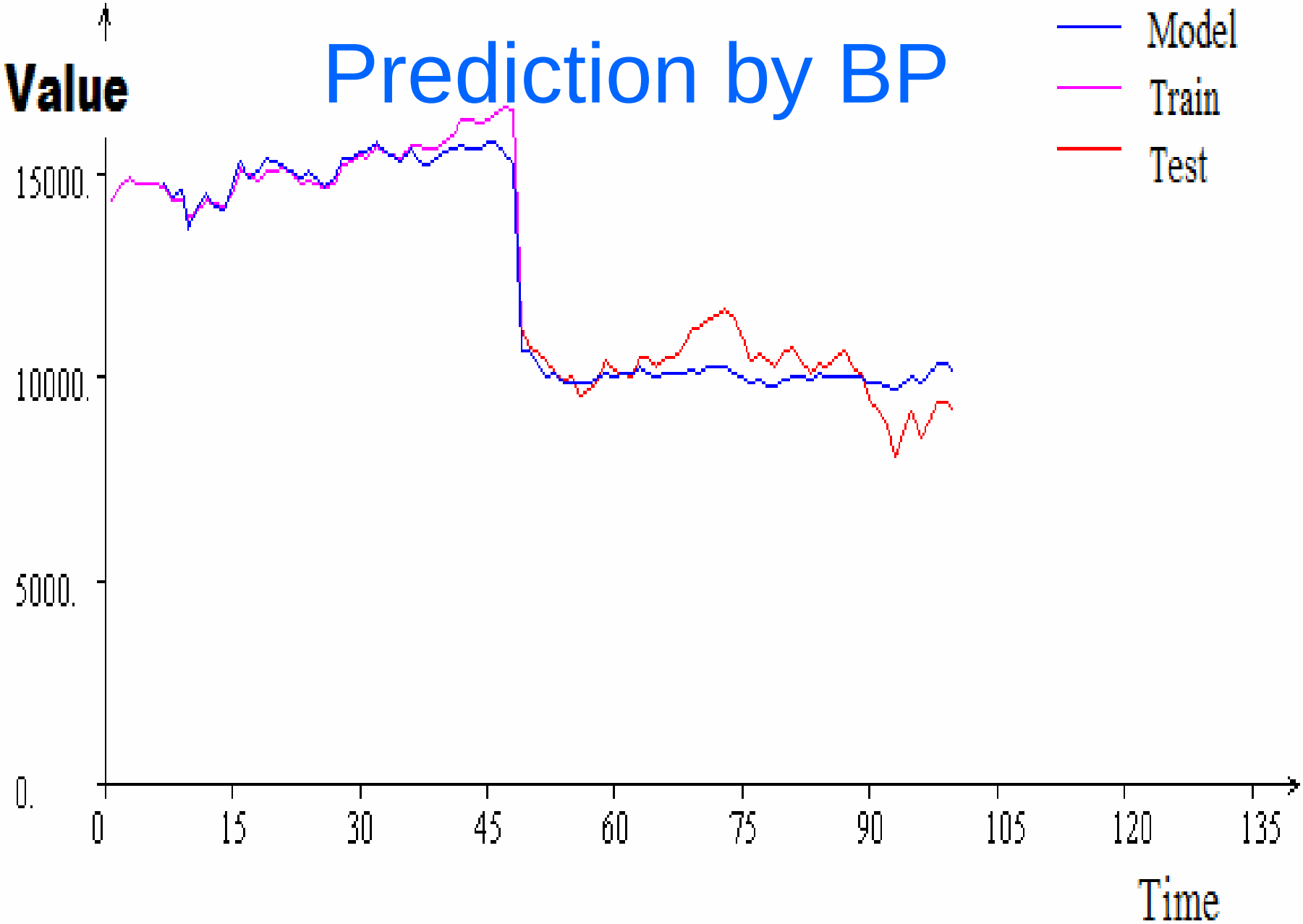
3. Compute outputs O, adjust weights according to the delta rule, backpropagating the errors. The weights will be nudged closer so that the network learns to give the desired output.

4. Repeat; stop when no errors, or enough epochs completed

Backpropagation

- Very powerful - can learn any function, given enough hidden units! With enough hidden units, we can generate any function.
- Have the same problems of Generalization vs. Memorization. With too many units, we will tend to memorize the input and not generalize well. Some schemes exist to “prune” the neural network.
- Networks require extensive training, many parameters to fiddle with. Can be extremely slow to train. May also fall into local minima.
- Inherently parallel algorithm, ideal for multiprocessor hardware.
- Despite the cons, a very powerful algorithm that has seen widespread successful deployment.

Prediction by BP



内容提要

- 引言
- 感知器
- 反传**Back Propagation**
- 递归网络**Recurrent network**
- **Hopfield** 网络
- 自组织映射
- 讨论和展望

递归网络

- A sequence is a succession of patterns that relate to the same object.
- For example, letters that make up a word or words that make up a sentence.
- Sequences can vary in length. This is a challenge.
- How many inputs should there be for varying length inputs ?

简单递归网络

- Jordan network has connections that feed back from the output to the input layer and also some input layer units feed back to themselves.
- Useful for tasks that are dependent on a sequence of a successive states.
- The network can be trained by backpropogation.
- The network has a form of short-term memory.
- Simple recurrent network (SRN) has a similar form of short-term memory.

Jordan递归网络

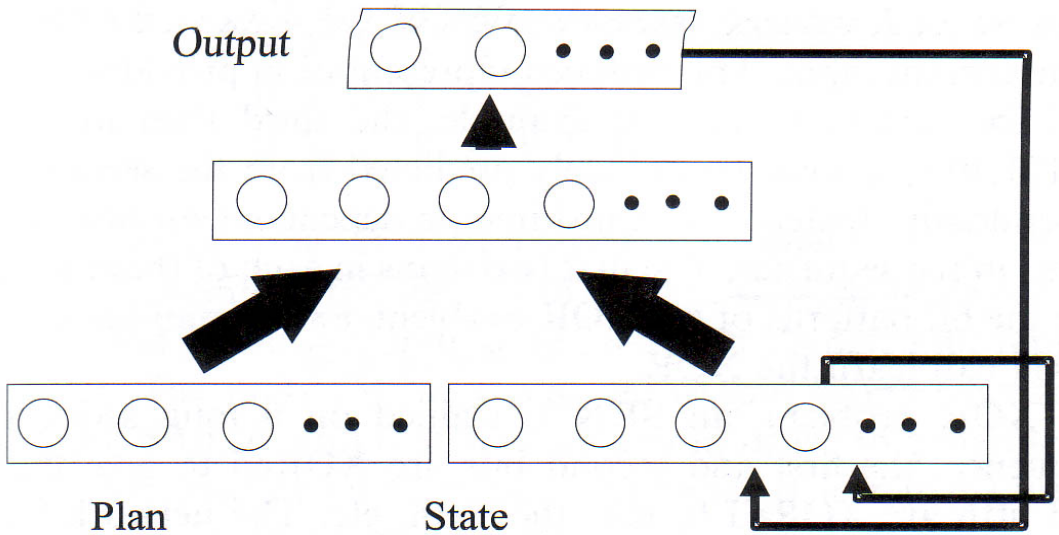


Figure 5.6 Jordan network.

Elman递归网络 (SRN)

The number of context units is the same as the number of hidden units

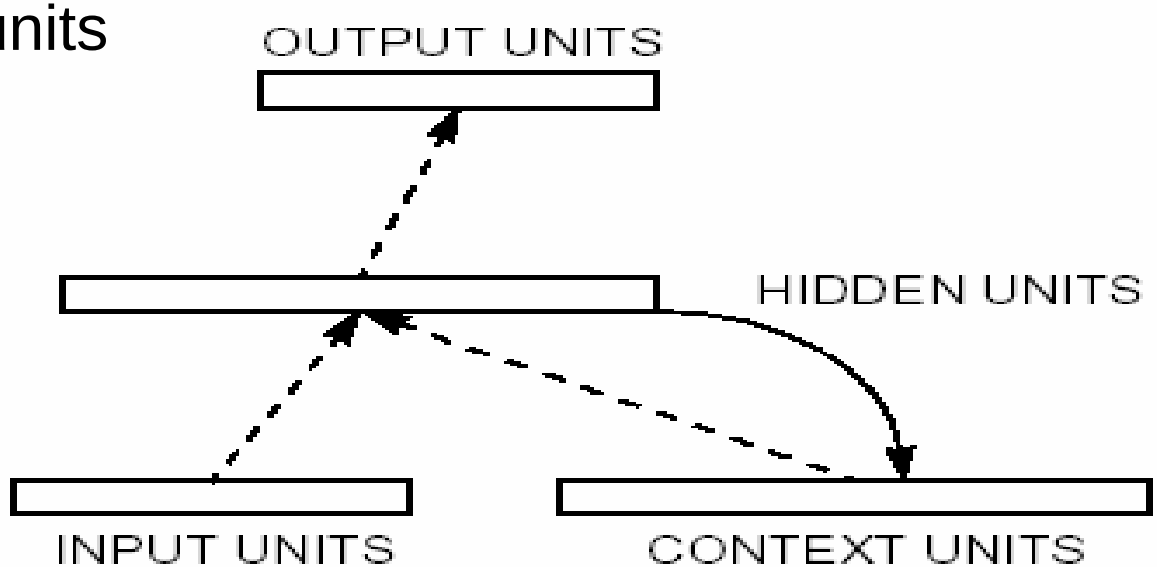


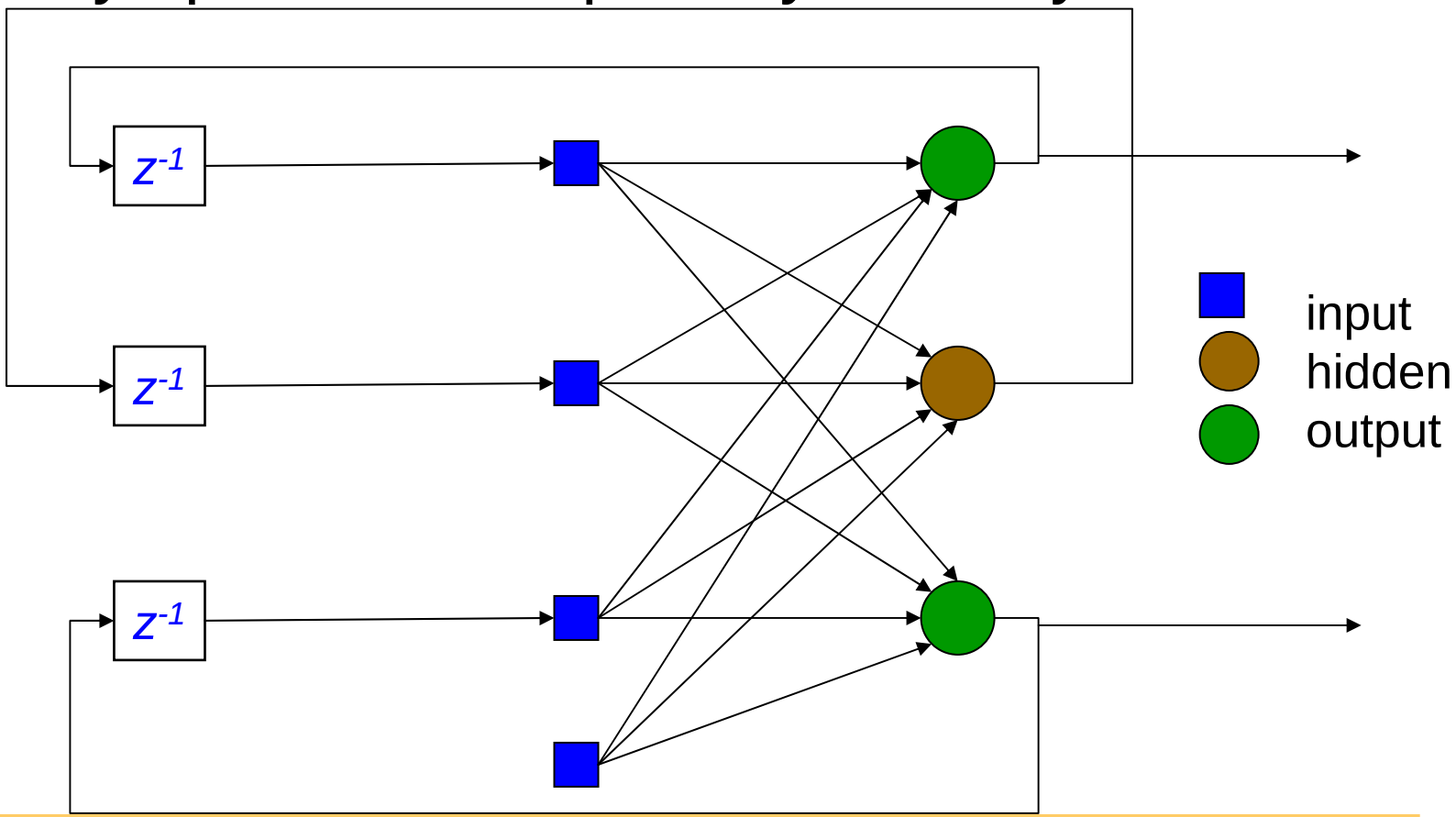
Figure 2. A simple recurrent network in which activations are copied from hidden layer to context layer on a one-for-one basis, with fixed weight of 1.0. Dotted lines represent trainable connections.

SRN递归网络记忆

- The context units remember the previous internal state.
- Thus, the hidden units have the task of mapping both an external input and also the previous internal state to some desired output.

递归网络

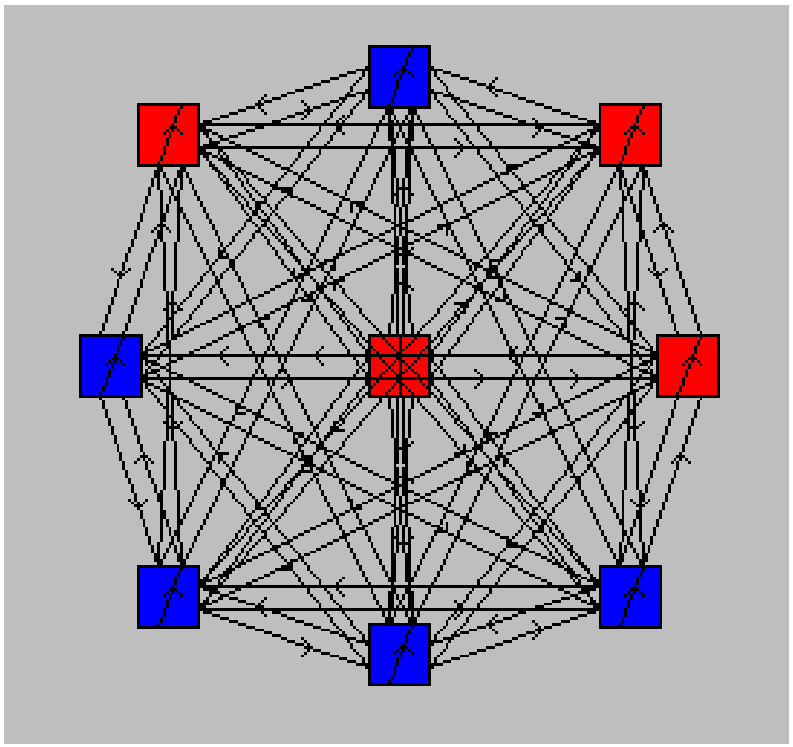
Recurrent Network with ***hidden neuron(s)***: unit delay operator z^{-1} implies dynamic system



内容提要

- 引言
- 感知器
- 反传**Back Propagation**
- 递归网络**Recurrent network**
- **Hopfield** 网络
- 自组织映射
- 讨论和展望

Hopfield 网络



- 1982年，Hoopfield用能量函数的思想形成一种具有对称连接的递归网络所执行的计算的新方法。这类具有反馈的特殊神经网络在80年代引起了大量的关注，产生了著名的Hopfield网络。尽管Hopfield网络不可能是真正的神经生物系统模型，他们包涵的原理，即在动态的稳定网络中存储信息原理的，是极深刻的。

Hopfield 网络

- John Hopfield (1982)
  - Associative Memory via artificial neural networks
  - Solution for optimization problems
  - Statistical mechanics

线性联想记忆

- 假設有一組輸入/輸出對，$(\underline{x}_i, \underline{y}_i), i = 1, \cdots, N$ ，$\underline{x}_i$ 與 $\underline{y}_i$ 的維度分別爲 p 和 r。我們的目標是想將這 N 個輸入/輸出對，用聯想記憶的方式予以儲存起來，也就是說，當輸入爲 $\underline{x}_i$或是加了一些雜訊的 $\underline{x}_i+\underline{n}_i$ (代表雜訊)，輸出會是 $\underline{y}_i$ 。

$$W = \begin{bmatrix} w_{1,1} & \cdots & w_{1,p} \\ \vdots & \ddots & \vdots \\ w_{r,1} & \cdots & w_{r,p} \end{bmatrix} = \sum_{k=1}^{N} \underline{y}_k \underline{x}_k^T$$

- 一、網路學習

$$w_{ji}(k+1) = w_{ji}(k) + \eta x_{ki} y_{kj}$$

- 二、網路回想

$$\underline{y} = W\underline{x}_k$$

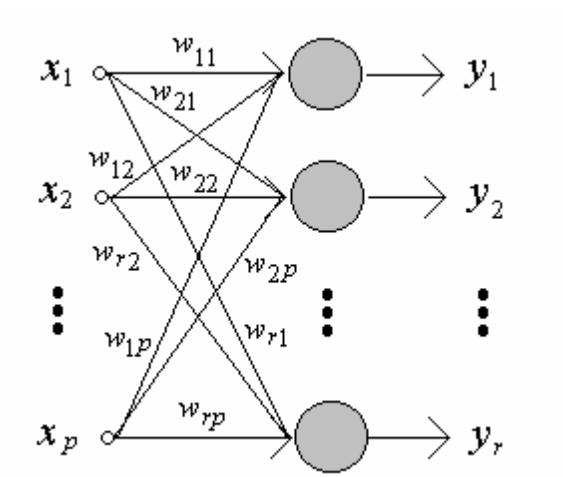


圖5.1爲實現線性聯想記憶的類神經網路示意圖。

线性联想记忆

- 如果輸入彼此之間爲正規化正交 (orthonormal)，即

$$\underline{x}_i^T \underline{x}_j = \begin{cases} 1 & \text{若 } i = j \\ 0 & \text{若 } i \neq j \end{cases}$$

- 假設現在的輸入爲 $\underline{x}_k$，則網路的輸出爲：

$$\underline{y} = W\underline{x}_k = \underline{y}_k \underline{x}_k^T \cdot \underline{x}_k + \sum_{i \neq k} \underline{y}_i \underline{x}_i^T \cdot \underline{x}_k = \underline{y}_k$$

- 假設 $\underline{x}_k$ 只是長度爲 1 的單位向量，那麼網路輸出可重寫爲：

$$\underline{y} = \underline{y}_k \underline{x}_k^T \cdot \underline{x}_k + \sum_{i \neq k} \underline{y}_i \underline{x}_i^T \cdot \underline{x}_k = \underline{y}_k + \underline{\Delta}_k$$

其中 $\underline{\Delta}_k$ 是所謂的 “雜訊向量 (noise vector)” 或稱爲 “交互影響 (crosstalk)”。

- 如果鍵結值矩陣是由下式計算而得的，那麼這些雜訊向量 $\underline{\Delta}_k$ 會降到最小：

$$W = YX^+ = YX^T (XX^T)^{-1}$$

$$Y = [\underline{y}_1, \cdots, \underline{y}_N] \qquad X = [\underline{x}_1, \cdots, \underline{x}_N]$$

其中 X^+ 稱爲 X 的虛擬反矩陣。

线性联想记忆

- 假設我們想要儲存以下三個輸入/輸出對：$(\underline{x}_1, \underline{y}_1), (\underline{x}_2, \underline{y}_2), (\underline{x}_3, \underline{y}_3)$

$$\underline{x}_1 = (1,0,0)^T, \underline{x}_2 = (0,1,0)^T, \underline{x}_3 = (0,0,1)^T$$

$$\underline{y}_1 = (2,1,2)^T, \underline{y}_2 = (1,2,3)^T, \underline{y}_3 = (3,1,2)^T$$

- 經由前述之公式可知

$$W = \underline{y}_1 \underline{x}_1^T + \underline{y}_2 \underline{x}_2^T + \underline{y}_3 \underline{x}_3^T$$

$$= \begin{pmatrix} 2 \\ 1 \\ 2 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \begin{pmatrix} 0 & 1 & 0 \end{pmatrix} + \begin{pmatrix} 3 \\ 1 \\ 2 \end{pmatrix} \begin{pmatrix} 0 & 0 & 1 \end{pmatrix}$$

- 如果網路現在的輸入是 $\underline{x}_2 = (0,1,0)$，那麼網路的輸出爲

$$\underline{y} = W \cdot \underline{x}_2 = \begin{pmatrix} 2 & 1 & 3 \\ 1 & 2 & 1 \\ 2 & 3 & 2 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} = \underline{y}_2$$

Hopfield 网络

- 如果輸入及輸出向量都是由二元值（0與1）或雙極值（-1與1）所構成的向量，那麼式 (5.8) 的輸出雜訊向量可以藉助下述的非線性轉換技術達到抑制雜訊的效果：

  (1) 將輸出取閾值 (thresholding) 以及

  (2) 將取閾值後的輸出，轉輸入回網路，然後重覆這兩項步驟直到收斂爲止。

- 這種有回授的類神經網路被稱爲“循環類神經網路 (recurrent neural networks)”，而離散的 Hopfield 網路 (discrete Hopfield networks) 便是此種網路的代表。

Hopfield 网络

一、網路學習 :假設有 N 筆輸入向量 $\underline{x}_i = [x_{i1}, \cdots, x_{ip}]^T, i = 1, \cdots, N$ 要用自聯想的方式儲存至離散 Hopfield 網路上

$$w_{ji} = \frac{1}{p} \sum_{k=1}^{N} x_{ki} x_{kj}$$

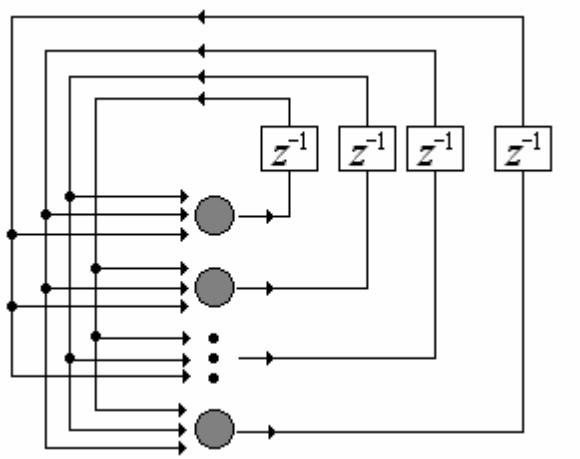
$$\theta_j = 0,\ j = 1, \cdots, p$$

$$W = \begin{bmatrix} w_{11} & \cdots & w_{1p} \\ \vdots & \ddots & \vdots \\ w_{p1} & \cdots & w_{pp} \end{bmatrix}$$

- 或者

$$\theta_j = \sum_{i=1}^{p} w_{ji},\ i = 1, \cdots, p$$

$$= \frac{1}{p} \sum_{k=1}^{N} \underline{x}_k \underline{x}_k^T - \frac{N}{p} I$$



Hopfield 网络

二、網路回想

- 假若此時有一輸入 $\underline{x}$ 進入此網路，我們將此時的輸入視做網路的初始輸出 $\underline{x}(0)$，緊接著，每個類神經元的後續輸出是由下式計算

$$x_j(n+1) = \mathrm{sgn}\left(\sum_{i=1}^{p} w_{ji} x_i(n) - \theta_j\right) = \mathrm{sgn}(u_j(n) - \theta_j)$$

$$= \begin{cases} 1 & \text{若} u_j(n) > \theta_j \\ x_j(n) & \text{若} u_j(n) = \theta_j \\ -1 & \text{若} u_j(n) < \theta_j \end{cases}$$

- 離散 Hopfield 網路是用非同步 (asynchronization) 的方式來變更類神經元的輸出，整個聯想過程可以用下列之鏈狀關係來描述：

$$\underline{x}(0) \to \underline{x}(1) \to \underline{x}(2) \to \cdots \to \underline{x}(k) \to \underline{x}(k+1) \to \cdots$$

- 如果我們用同步 (synchronization) 的方式來變更網路輸出，結果會有些不同

离散Hopfield 网络

- 如果類神經元輸出的更新是採用非同步模式，則網路必定會收斂至某一穩定狀態。
- 動態系統的穩定度分析，最直接的方式是解出系統方程式分析。
- Hopfield 網路是否穩定通常採用 Lyapunov 法 。
- 首先，我們定義離散 Hopfield 網路的能量函數爲：

$$E(k) = -\frac{1}{2}\underline{x}^T(k)W\underline{x}(k) + \underline{x}^T(k)\underline{\theta} = -\frac{1}{2}\sum_{h=1}^{p}\sum_{j=1}^{p} w_{jh}x_h(k)x_j(k) + \sum_{h=1}^{p}\theta_h x_h(k)$$

- 假設第 *i* 個類神經元的輸出於時間 *k* 時產生了變化，即

$$\Delta x_i(k) = x_i(k+1) - x_i(k)$$

- 由於 $\Delta x_i(k)$ 的產生，導致網路的能量於時間 *k*+1 時變成

$$E(k+1) = \left[-\frac{1}{2}\sum_{h\neq i}\sum_{j\neq i} w_{hj}x_h(k)x_j(k) + \sum_{h\neq i}\theta_h x_h(k)\right]$$

$$+\left[-\frac{1}{2}\sum_{j\neq i} w_{ij}x_i(k+1)x_j(k) - \frac{1}{2}\sum_{h\neq i} w_{hi}x_h(k)x_i(k+1) + \theta_i x_i(k+1)\right]$$

离散Hopfield 网络

- 因此，網路能量的變化量爲

$$\Delta E = E(k+1) - E(k)$$

$$= -\frac{1}{2}\sum_{j\neq i} w_{ij}\left(x_i(k+1)x_j(k) - x_i(k)x_j(k)\right)$$

$$-\frac{1}{2}\sum_{h\neq i} w_{hi}\left(x_h(k)x_i(k+1) - x_h(k)x_i(k)\right)$$

$$+\theta_i\left(x_i(k+1) - x_i(k)\right)$$

- 接著，我們將下標 h 改爲下標 j，又因爲 $w_{ij}=w_{ji}$，因此可以將上式爲

$$\Delta E = -\sum_{j\neq i} w_{ij}\left[x_i(k+1)x_j(k) - x_i(k)x_j(k)\right] + \theta_i\left[x_i(k+1) - x_i(k)\right]$$

$$= -\left[x_i(k+1) - x_i(k)\right]\left[\sum_{j\neq i} w_{ij}x_j(k) - \theta_i\right]$$

$$= -\Delta x_i(k)\cdot\left[\sum_{j\neq i} w_{ij}x_j(k) - \theta_i\right]$$

离散Hopfield 网络

由於 $\Delta x_i(k)$ 的值只有三種可能，現在我們就分三種情況來分析 ΔE

1. $\Delta x_i(k) = 2$：即 $x_i(k+1) = 1$ 且 $x_i(k) = -1$

   $x_i(k+1) = 1$代表 $\sum_{j \neq i} w_{ij} x_j(k) - \theta_i > 0$ ，因此 $\Delta E<0$

2. $\Delta x_i(k) = -2$：即 $x_i(k+1) = -1$ 且 $x_i(k) = 1$

   $x_i(k+1) = -1$代表 $\sum_{j \neq i} w_{ij} x_j(k) - \theta_i < 0$ ，因此 $\Delta E<0$

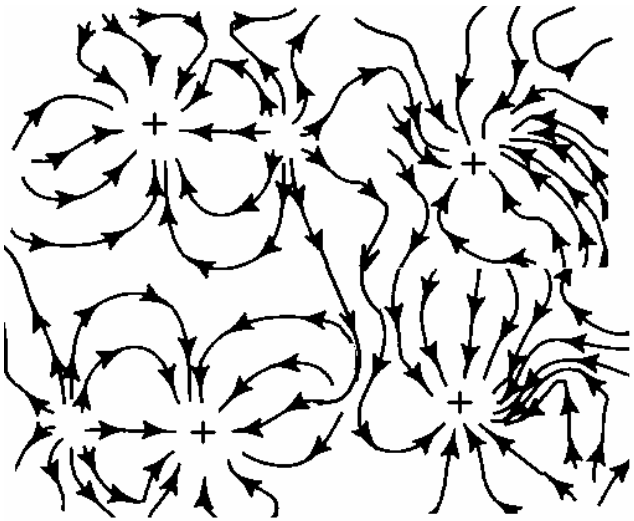
3. $\Delta x_i(k) = 0$：即 $x_i(k+1) = x_i(k)$

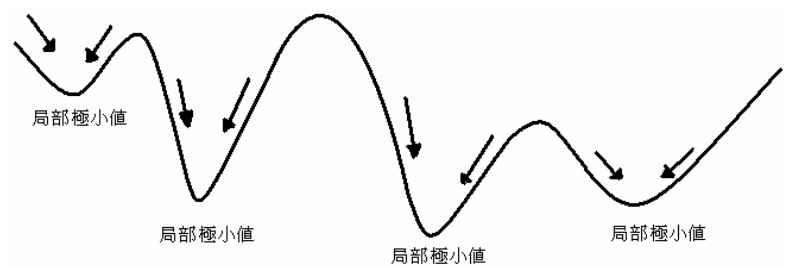
   $x_i(k+1) = x_i(k)$代表 $\sum_{j \neq i} w_{ij} x_j(k) - \theta_i = 0$ ，因此 $\Delta E=0$

綜合上列三種情況，我們得知 ***$\Delta E<=0$***

离散Hopfield 网络

- 離散 Hopfield 網路利用能量函數的局部極小特性來儲存資料。
- 我們利用 Hebbian 規則來調整網路的鍵結值，有可能會使得網路能量之局部極小值的數目會超過原先儲存的資料數目，也就是會有"偽造狀態(Spurious states)"的產生。





- 離散 Hopfield 網路的記憶容量有其上限，若類神經元的數目是 p，在記憶提取有 99% 正確率的情況下，則可儲存的資料筆數不會超過下式。

$$\text{記憶容量} \leq \frac{p}{4\ln p}$$

连续离散Hopfield 网络

- Hopfield 於 1984 年提出了連續 Hopfield 網路 (analog or continuous Hopfield networks) [11]。網路中的類神經元採用 S 曲線的活化函數，網路的狀態以及輸出可以用下列之微分方程式來描述：

$$\begin{cases} C_i \dfrac{du_i}{dt} = \sum_{j \neq i} w_{ij} y_j - \dfrac{u_i}{R_i} + I_i \\ y_i = \varphi(u_i) \end{cases}$$

其中，對第 i 個類神經元來說，C_i 代表細胞膜的電容性，R_i 是細胞膜的電阻性，I_i 是外界輸入電流(代表閥值項)，y_i 是類神經元的輸出電位差，u_i 代表軸突丘的細胞膜電位差，以及 w_{ij} 代表第 j 個類神經元至第 i 個類神經元的鍵結值，$\varphi(.)$ 是個嚴格遞增的 S-型活化函數。

连续离散Hopfield 网络

- 根據克希荷夫電流定律（Kirchhoff equation）

$$C_i \frac{du_i}{dt} + \frac{u_i}{\rho_i} = \sum_{j \neq i} \frac{1}{R_{ij}} (y_j - u_i) + I_i$$

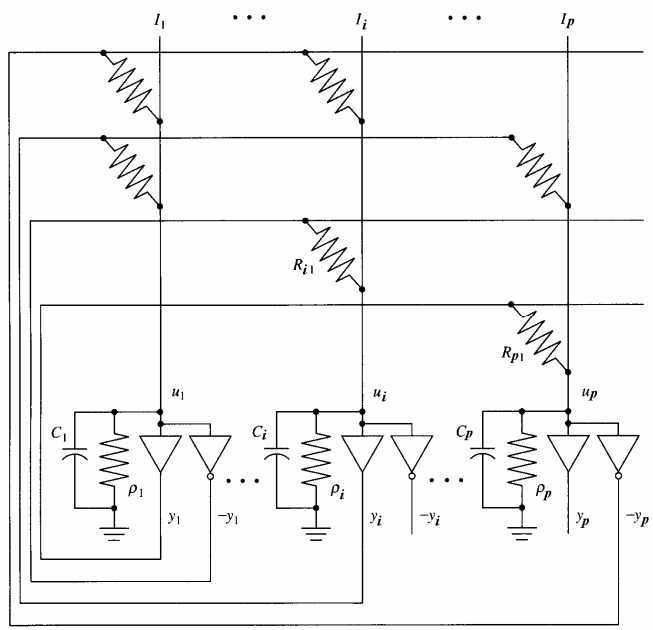
其中　代表非線性放大器的輸入電阻性。如果我們定義

$$\begin{cases} \dfrac{1}{R_i} = \dfrac{1}{\rho_i} + \sum_j \dfrac{1}{R_{ij}} \\ w_{ij} = \dfrac{1}{R_{ij}} \end{cases}$$

- 可以將式轉換成如下

$$C_i \frac{du_i}{dt} = \sum_j w_{ij} y_j - \frac{u_i}{R_i} + I_i$$

- 上式再加上　就等同於式



Hopfield 网络的应用

- Hopfield 網路的應用範圍很廣，譬如說，有些人把它應用於類比/數位轉換器的設計。
- 另一個廣受注意的應用就是用來解決优化的問題，主要的關鍵技術就是想辦法將欲最佳化的目標函數（object function）整理成 Hopfield 網路的 Lyapunov 函數，然後便可發現鍵結值應如何設定，最後我們初始化網路的輸出值，然後隨著時間的變化，網路會收斂至穩定狀態，而此穩定狀態便是目標函數的極小值，在此強調一點，我們無法保證它是全域極小值（global minima）。

推销员问题 (TSP)

- 如何設計一條最短的行程以便走過 N 個城市，而每個城市只能去一次，並且最後要回到原先出發的城市。
- 類神經元被安排成 $N \times N$ 的矩陣，行（column）代表旅途的順序，列（row）代表城市名稱，而矩陣的每一個元素 — 亦即類神經元的輸出，都是單極性的二元值，若矩陣的第　元素為 1，則代表城市 y 是第 i 站。
- 圖5.6顯示這麼一條路徑：B → D → A → C → (B)。

	1	2	3	4
A	0	0	1	0
B	1	0	0	0
C	0	0	0	1
D	0	1	0	0

圖5.6：推銷員旅途長度問題的解答之一：B → D → A → C → (B)。

推销员问题(TSP)

(1) 每個城市只能去一次：

$$E_1 = \sum_{y=1}^{N} \sum_{i=1}^{N} \sum_{j \neq i}^{N} x_{y,i} x_{y,j}$$

(2) 一次只能去一個城市：

$$E_2 = \sum_{i=1}^{N} \sum_{y=1}^{N} \sum_{\substack{z=1 \\ z \neq y}}^{N} x_{y,i} x_{z,i}$$

(3) 所有城市都要被經過：

$$E_3 = \left(\sum_{y=1}^{N} \sum_{i=1}^{N} x_{y,i} - N \right)^2$$

(4) 路途距離最短：

$$E_4 = \sum_{y=1}^{N} \sum_{\substack{z=1 \\ z \neq y}}^{N} \sum_{i=1}^{N} d_{y,z} x_{y,i} (x_{z,i+1} + x_{z,i-1})$$

其中 代表城市 y 與城市 z 的距離。

推销员问题 (TSP)

- 現在我們想要同時將 E_1、E_2、E_3 以及 E_4 最小化，因此可以將上述四個函數乘上不同的權值，以便合併考慮：

$$E = w_1 E_1 + w_2 E_2 + w_3 E_3 + w_4 E_4 \qquad (36)$$

- 我們將含有 $N \times N$ 個類神經元的 Hopfield 網路的能量函數寫於下式：

$$E = -\frac{1}{2} \sum_{y=1}^{N} \sum_{i=1}^{N} \sum_{z=1}^{N} \sum_{j=1}^{N} w_{(y,i),(z,j)} x_{y,i} x_{z,j} + \sum_{y=1}^{N} \sum_{i=1}^{N} \theta_{y,i} x_{y,i} \qquad (37)$$

- 比較式(36)與式(37)，我們便可知道鍵結值 $w_{(y,i),(z,j)}$（第(y, i)個類神經元與第(z, j)個類神經元的聯結鍵結值）以及閥值$\theta_{(y,i)}$應照下列式子設定：

$$w_{(y,i),(z,j)} = -2w_1\delta_{y,z}(1-\delta_{i,j}) - 2w_2\delta_{i,j}(1-\delta_{y,z}) - 2w_3 - 2w_4 d_{y,z}(\delta_{j,i+1} + \delta_{j,i-1})$$

$$\theta_{y,i} = -2w_3 N$$

$$\delta_{y,z} \triangleq \begin{cases} 1 & \text{如果} y = z \\ 0 & \text{如果} y \neq z \end{cases}$$

推销员问题(TSP)

- 假設城市數目N=4，所以鍵結值矩陣是個16×16的矩陣（因為共有16個類神經元在這4×4的Hopfield網路），$d_{A,B} = d_{B,C} = d_{C,D} = d_{D,A} = 1$ km，$d_{A,C} = d_{B,D} = \sqrt{2}$ km，並且讓$w_1 = w_2 = w_3 = w_4 = 0.6$，我們得到：

$$W_{16\times16} = [w_{(y,i),(z,j)}] = \begin{bmatrix} -1.2 & \cdots & -2.4 \\ \vdots & \ddots & \vdots \\ -2.4 & \cdots & -1.2 \end{bmatrix} \qquad \underline{\theta} = (\theta_1, \theta_2, \cdots, \theta_{16})^T = (-8, -8, \cdots, -8)^T$$

$$w_{16,16} = w_{(4,4),(4,4)}$$

$$= -2\cdot0.6\delta_{4,4}(1-\delta_{4,4}) - 2\cdot0.6\delta_{4,4}(1-\delta_{4,4}) - 2\cdot0.6 - 2\cdot0.6\cdot(\delta_{4,5}+\delta_{4,3}) = -1.2$$

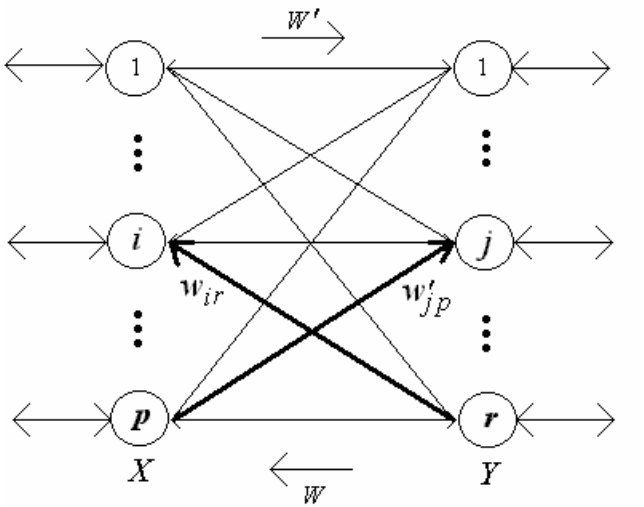
0	1	0	0
0	0	1	0
1	0	0	0
0	0	0	1

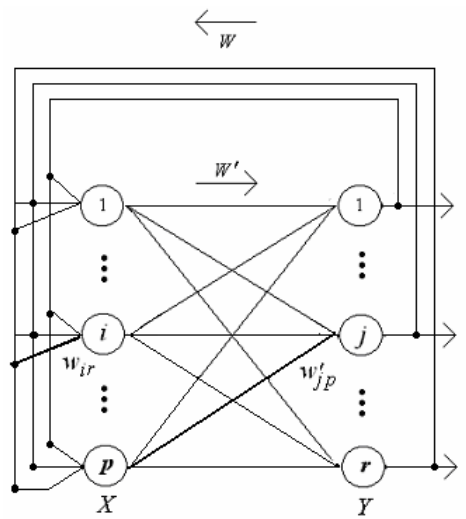
如果我們採用同步疊代方式來更新網路狀態，那麼根據式 ，網路的新狀態變成

0	1	0	0
0	0	1	0
1	0	0	0
0	0	0	1

双向联想记忆

- 雙向聯想記憶（bidirectional associative memory, 簡稱 BAM）是一種執行異聯想（heteroassociation）的雙層類神經網路。
- 它可被用來儲存M個輸入/輸出向量對，$(\underline{x}_i, \underline{y}_i)$，$i=1,...M$，$\underline{x}_i$ 與 $\underline{y}_i$ 分別為 p 維與 r 維的向量，其基本架構如圖5.7所示。
- 它利用順反兩個方向的資訊傳輸方式，使得網路的兩層類神經元的輸出，以一種反覆出現的模式達到穩定狀態，藉此完成異聯想的工作。





双向联想记忆

- 假設 Y 層的類神經元在時間k時的輸出為 $\underline{y}(k)$，這時 $\underline{y}(k)$ 便被視為位於 X 層的類神經元的輸入，然後這些類神經元（ X 層）便根據下式來更新類神經元的輸出：

$$\underline{x}(k+1) = \varphi(W\underline{y}(k) - \underline{\theta})$$

$$x_i(k+1) = \varphi\left(\sum_{j=1}^{r} w_{ij} y_j(k) - \theta_i\right), \quad i = 1, \cdots, p$$

$$= \begin{cases} +1 & \text{如果} \sum_{j=1}^{r} w_{ij} y_j(k) > \theta \\ x_i(k-1) & \text{如果} \sum_{j=1}^{r} w_{ij} y_j(k) = \theta \\ -1 & \text{如果} \sum_{j=1}^{r} w_{ij} y_j(k) < \theta \end{cases}$$

其中

$$\underline{x}(k+1) = [x_1(k+1), \ldots, x_p(k+1)]^T, \quad \underline{y}(k) = [y_1(k), \ldots, y_r(k)]^T,$$

双向联想记忆

- 然後將 $\underline{x}(k+1)$ 視為位於 Y 層的類神經元的輸入，並根據下式來計算類神經元（Y 層）新的輸出：

$$\underline{y}(k+2)=\varphi(W'\underline{x}(k+1)-\underline{\theta'})$$

$$y_j(k+2)=\varphi(\sum_{i=1}^{p}w'_{ji}x_i(k+1)-\theta'_j)\quad j=1,2,\cdots,r$$

- 這個過程會一直反覆持續下去，直到收斂為止，整個回想過程可以用以下的步驟來總結。

第 1 次反饋傳輸：	$\underline{x}(1)=\varphi(W\underline{y}(0)-\underline{\theta})$
第 1 次前饋傳輸：	$y(2)=\varphi(W'\underline{x}(1)-\underline{\theta'})$
第 2 次反饋傳輸：	$\underline{x}(3)=\varphi(W\underline{y}(2)-\underline{\theta})$
第 2 次前饋傳輸：	$y(4)=\varphi(W'\underline{x}(3)-\underline{\theta'})$
…	…
第 $k/2$ 次反饋傳輸：	$\underline{x}(k-1)=\varphi(W\,y(k-2)-\underline{\theta})$
第 $k/2$ 次前饋傳輸：	$\underline{y}(k)=\varphi(W'\underline{x}(k-1)-\underline{\theta'})$
…	…

双向联想记忆

- 採用 Hebbian 學習規則

$$W=\sum_{k=1}^{N}\underline{x}_k\underline{y}_k^T \qquad w_{ij}=\sum_{k=1}^{N}x_i(k)y_j(k)\quad i=1,\ldots,p\ \ j=1,\ldots r$$

$$W'=\sum_{k=1}^{N}\underline{y}_k\underline{x}_k^T \qquad w'_{ji}=\sum_{k=1}^{N}y_j(k)x_i(k)\quad i=1,\ldots p\ \ j=1,\ldots,r$$

$$\theta_i=\frac{1}{2}\sum_{j=1}^{r}w_{ij} \qquad \theta_i=0$$

$$\theta'_j=\frac{1}{2}\sum_{i=1}^{p}w'_{ji} \qquad \theta_j'=0$$

- 我們可以看出 $W^T=W$，亦即 $w'_{ji}=w_{ij}$。
- 所謂的雙向穩定就是網路經過多次雙向疊代之後，會有以下的情況發生：

$$\underline{x}(k)\rightarrow\underline{y}(k+1)\rightarrow\underline{x}(k+2)\rightarrow\underline{y}(k+3)\rightarrow\cdots\rightarrow$$

其中

$$\underline{x}(k)=\underline{x}(k+2) \qquad \underline{y}(k+1)=\underline{y}(k+3)$$

双向联想记忆

- 網路的 Lyapunov 函數為（為了簡化問題，我們令閥值項 ）：

$$\underline{\theta} = \underline{\theta'} = \underline{0}$$

$$E(\underline{x}, \underline{y}) = -\frac{1}{2}\underline{x}^T W^T \underline{y} - \frac{1}{2}\underline{y}^T W \underline{x} = -\underline{y}^T W \underline{x}$$

- 我們發現更動 或 中的任一個位元資訊(例如從 +1 → −1 或從 −1 → +1)，會導致網路能量變化如下：

$$\Delta E_{x_i} = -\left(\sum_{j=1}^{r} w_{ij} y_j\right)\Delta x_i, \quad i = 1, \cdots, p$$

$$\Delta E_{y_j} = -\left(\sum_{i=1}^{n} w'_{ji} x_i\right)\Delta y_j, \quad j = 1, \cdots, r$$

双向联想记忆

- 而 與 的變化量不外乎以下三種:

$$\Delta x_i = \begin{cases} 2 & \text{如果} \sum_{j=1}^{r} w_{ij} y_j > 0 \\ 0 & \text{如果} \sum_{j=1}^{r} w_{ij} y_j = 0, \quad i = 1, \cdots, p \\ -2 & \text{如果} \sum_{j=1}^{r} w_{ij} y_j < 0 \end{cases} \qquad \Delta y_j = \begin{cases} 2 & \text{如果} \sum_{i=1}^{p} w'_{ji} x_i > 0 \\ 0 & \text{如果} \sum_{i=1}^{p} w'_{ji} x_i = 0, \quad j = 1, \cdots, r \\ -2 & \text{如果} \sum_{i=1}^{p} w'_{ji} x_i < 0 \end{cases}$$

- 我們很輕易地發現， Δx_i 和 $\sum_{j=1}^{r} w_{ij} y_j$ 同號以及 Δy_j 和 $\sum_{i=1}^{p} w'_{ji} x_i$ 同號，再加上有一個負號，所以 $\Delta E_{x_i} \le 0$ 和 $\Delta E_{y_j} \le 0$ ，除此之外，$E(\underline{x}, \underline{y})$有下限（ $E(\underline{x}, \underline{y}) \ge -\sum_{i=1}^{p} \sum_{j=1}^{r} |w_{ij}|$ ），所以可以得到這個結論 — BAM 會隨著時間的演進而往網路能量低的方向移動，最後收斂至雙向穩定狀態，而此狀態便是能量函數的局部極小點。值得一提的是，BAM也會有所謂的"偽造狀態"的出現。

双向联想记忆

- 如果將離散 BAM 的差分狀態改變規則，如式(5.42)及式(5.44)改成下列的微分方程式，則會變成所謂的連續 BAM (continuous bidirectional associative memory)：

$$\begin{cases} \dfrac{dx_i}{dt} = -x_i + \sum\limits_{j=1}^{r} w_{ij}\varphi(y_j) + I_i \\ \dfrac{dy_j}{dt} = -y_j + \sum\limits_{i=1}^{p} w'_{ji}\varphi(x_i) + I_j{}' \end{cases}$$

其中，I_i 以及 I'_j 爲正值常數。

双向联想记忆

- 給定三個輸入/輸出對 $(\underline{x}_1, \underline{y}_1)$、$(\underline{x}_2, \underline{y}_2)$ 和 $(\underline{x}_3, \underline{y}_3)$

$$\underline{x}_1 = (1,-1,1,-1)^T, \underline{y}_1 = (-1,1,1)^T$$

$$\underline{x}_2 = (-1,1,1,1)^T, \underline{y}_2 = (-1,-1,1)^T$$

$$\underline{x}_3 = (1,1,-1,-1)^T, \underline{y}_3 = (1,1,-1)^T$$

$$W = \sum_{k=1}^{3} \underline{x}_k \underline{y}_k^T = \underline{x}_1 \underline{y}_1^T + \underline{x}_2 \underline{y}_2^T + \underline{x}_3 \underline{y}_3^T$$

- 所以

$$W = \begin{pmatrix} 1 & 3 & -1 \\ 1 & -1 & -1 \\ -3 & -1 & 3 \\ 1 & -1 & 1 \end{pmatrix}, W' = W^T = \begin{pmatrix} 1 & 1 & -3 & -1 \\ 3 & -1 & -1 & -3 \\ -1 & -1 & 3 & 1 \end{pmatrix}$$

双向联想记忆

- 假設目前的輸入向量 $\underline{x} = (1,-1,1,-1) = x_1$，根據式(5.41)及式(5.43)，我們可以得到下列之疊代情形：

$$\underline{x} = \begin{pmatrix} 1 \\ -1 \\ 1 \\ -1 \end{pmatrix} \rightarrow \underline{y} = \varphi \begin{pmatrix} -2 \\ 6 \\ 2 \end{pmatrix} = \begin{pmatrix} -1 \\ 1 \\ 1 \end{pmatrix} \rightarrow \underline{x} = \varphi \begin{pmatrix} 1 \\ -3 \\ 5 \\ -1 \end{pmatrix} = \begin{pmatrix} 1 \\ -1 \\ 1 \\ -1 \end{pmatrix} \rightarrow \underline{y} = \varphi \begin{pmatrix} -2 \\ 6 \\ 2 \end{pmatrix} = \begin{pmatrix} -1 \\ 1 \\ 1 \end{pmatrix} \rightarrow \cdots$$

- 假設目前的輸入向量是 $\underline{x}$ =(-1,1,-1,-1)^T，我們可以得到下列之兩種疊代情形：

1.令y_2的前一刻值是1 (即 φ(0)=1)

$$\underline{x} = \begin{pmatrix} -1 \\ 1 \\ -1 \\ -1 \end{pmatrix} \rightarrow \underline{y} = \varphi \begin{pmatrix} 4 \\ 0 \\ -4 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ -1 \end{pmatrix} \rightarrow \underline{x} = \varphi \begin{pmatrix} 5 \\ 1 \\ -1 \\ 3 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ -1 \\ -1 \end{pmatrix} \rightarrow \underline{y} = \varphi \begin{pmatrix} 6 \\ 6 \\ -6 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ -1 \end{pmatrix}$$

2. 令y_2的前一刻是-1 (即φ(0)=-1)

$$\underline{x} = \begin{pmatrix} -1 \\ 1 \\ -1 \\ -1 \end{pmatrix} \rightarrow \underline{y} = \varphi \begin{pmatrix} 4 \\ 0 \\ -4 \end{pmatrix} = \begin{pmatrix} 1 \\ -1 \\ -1 \end{pmatrix} \rightarrow \underline{x} = \varphi \begin{pmatrix} -1 \\ 3 \\ -5 \\ 1 \end{pmatrix} = \begin{pmatrix} -1 \\ 1 \\ -1 \\ 1 \end{pmatrix} \rightarrow \underline{y} = \varphi \begin{pmatrix} 2 \\ -6 \\ -2 \end{pmatrix} = \begin{pmatrix} 1 \\ -1 \\ -1 \end{pmatrix}$$

双向联想记忆

- 根據 Kosko [19] 本身的粗略估計，BAM 的記憶容量最大爲 $\min(p,r)$，但是經過進一步的分析 [20]，更保守一點的估計應該不會超過 $\sqrt{\min(p,r)}$。
- BAM 的狀態疊代方式，不管是用同步 (synchronization) 或非同步 (asynchronization) 的模式，都會收斂至雙向穩定狀態。
- BAM 除了可被用來記憶靜態 (static) 的向量對之外，亦可用來儲存動態的狀態變化 (dynamic state transitions)，譬如說，想要儲存以下的一連串狀態向量：

$$\underline{s}_1 \to \underline{s}_2 \to \cdots \to \underline{s}_N \to \underline{s}_1 \to \underline{s}_2 \to \cdots \to \underline{s}_N \to \underline{s}_1 \to \cdots$$

- 我們可以將此問題視爲異聯想的記憶問題，亦即要 BAM 記得以下之向量對： $(\underline{s}_1,\underline{s}_2),(\underline{s}_2,\underline{s}_3),\cdots,(\underline{s}_{N-1},\underline{s}_N),(\underline{s}_N,\underline{s}_1)$

$$W=\sum_{k=1}^{N-1}\underline{s}_{k+1}\underline{s}_k^T+\underline{s}_1\underline{s}_N^T$$

$$\underline{s}_N \to \underline{s}_{N-1} \to \cdots \to \underline{s}_2 \to \underline{s}_1 \to \underline{s}_N \to \cdots$$

$$\begin{cases}\underline{x}=\varphi(W\underline{y})\\ \underline{y}=\varphi(W\underline{x})\end{cases} \qquad \begin{cases}\underline{x}=\varphi(W^T\underline{y})\\ \underline{y}=\varphi(W^T\underline{x})\end{cases}$$

内容提要

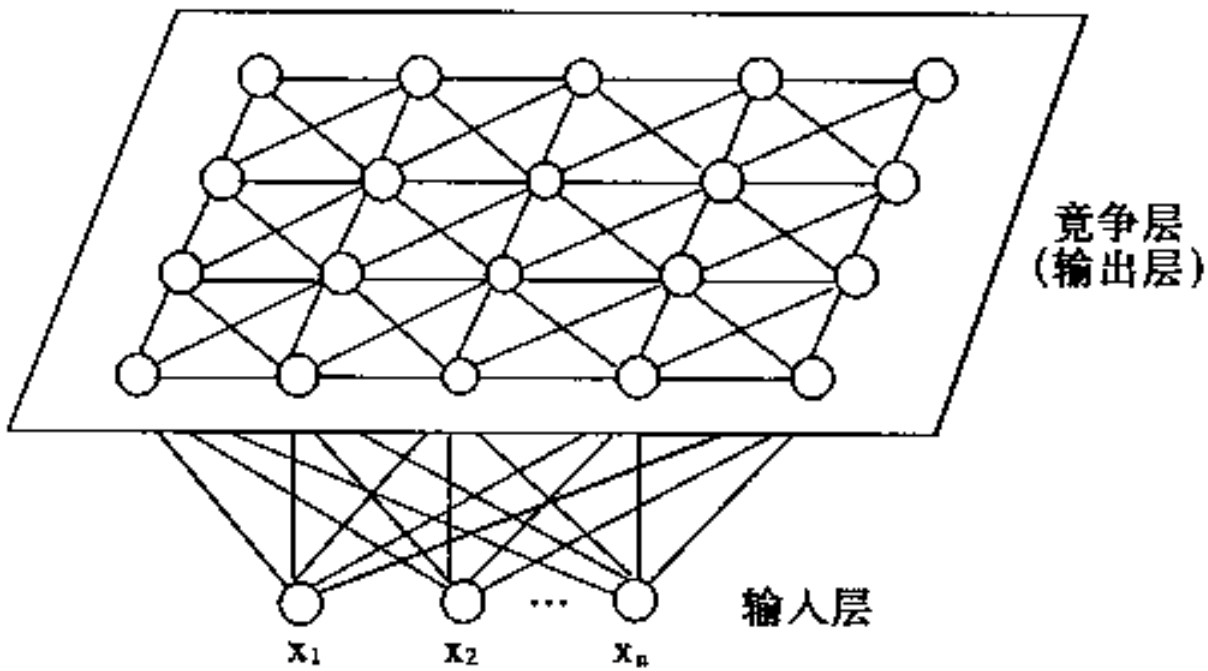
- 引言
- 感知器
- 反传**Back Propagation**
- 递归网络**Recurrent network**
- **Hopfield** 网络
- 自组织映射
- 讨论和展望

SOM简介

- Self-organizing map： 自组织映射
  是一种无指导训练的神经网络
- 由芬兰人Kohonen于1981年开始研究
  自组织的过程实际上就是一种无指导的学习。它通过自身训练，自动对输入模式进行聚类
- 在聚类、分类、机器人视觉、机械控制、语音识别、向量量化及组合优化等领域都有着广泛的应用。

SOM简介：拓扑结构

- 网络上层为输出结点(m个)，按二维形式排成一个结点矩阵
- 输入结点处于下方，若输入向量有n个元素，则输入端共有n个结点
- 所有的输入结点到所有的输出结点都有权值连接。



Kohonen网络训练算法

- 某个输出结点能对某一类模式作出特别的反应以代表该模式类
- 输出层上相邻的结点能对实际模式分布中相近的模式类作出特别的反映
- 当某类数据模式输入时，对某一输出结点产生最大刺激（获胜结点），同时对获胜结点周围的一些结点产生较大刺激。

Kohonen网络训练算法

- 定义获胜结点的邻域结点
- 在训练的过程中，不但对获胜结点的连接权值作调整，同时对获胜结点的邻域结点的连接权值作调整
- 随着训练的进行，这个邻域范围不但缩小，直到最后，只对获胜结点进行细微的连接权值调整。

Kohonen网络训练算法

1) 连接权值初始化。给从输入结点到输出结点的所有权值赋予较小的随机数。时间计数t=0

2) 对网络输入模式 $x^k=(x_1^k,x_2^k,\ldots,x_n^k)$

3) 计算输入x^k与全部输出结点所连的权向量的距离 $d_j=\sum_{i=1}^{n}(x_i^k-W_{ij})^2 \qquad j\in\{1,2,\ldots,m\}$

4) 具有最小距离的输出结点获胜：$d_{j^*}=\min_{j\in\{1,2,\ldots,m\}}\{d_j\}$

Kohonen网络训练算法

5) 调整输出结点 所连接的权值以及其邻域 N_{j^*}

$NE_{j^*}(t)$ 内的输出结点所连权值：

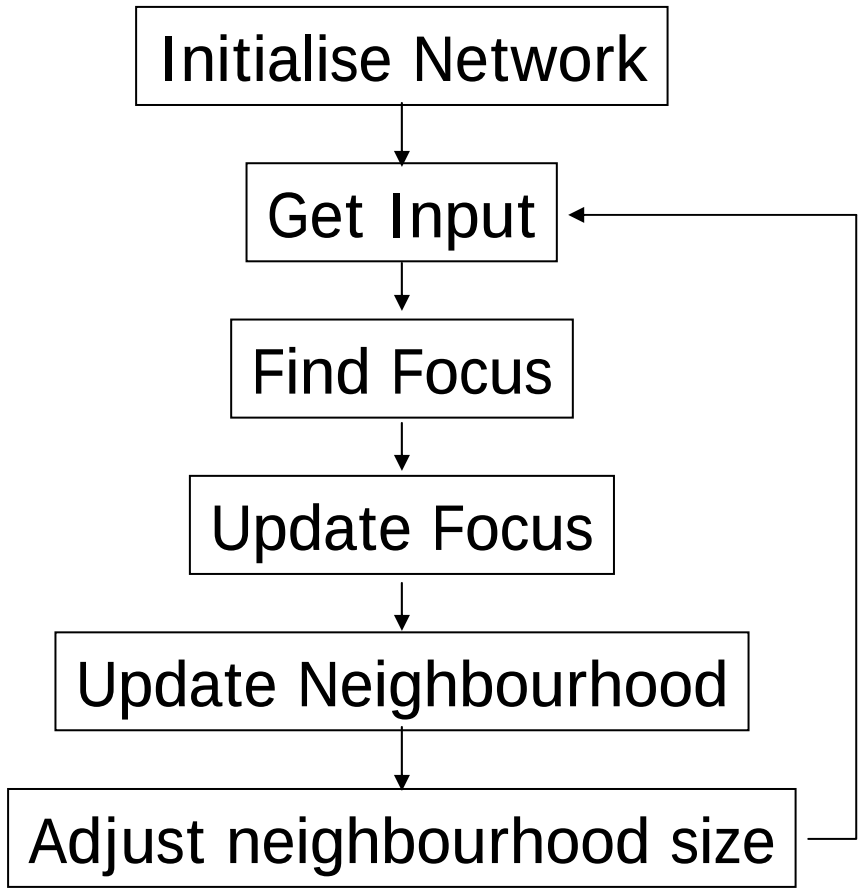
$$\Delta W_{ij} = \eta(t)(x_i^k - W_{ij}) \quad N_j \in NE_{j^*}(t) \quad i \in \{1,2,\ldots,n\}$$

6) 若还有输入样本数据，则t=t+1，转第2）步。

Kohonen网络训练算法

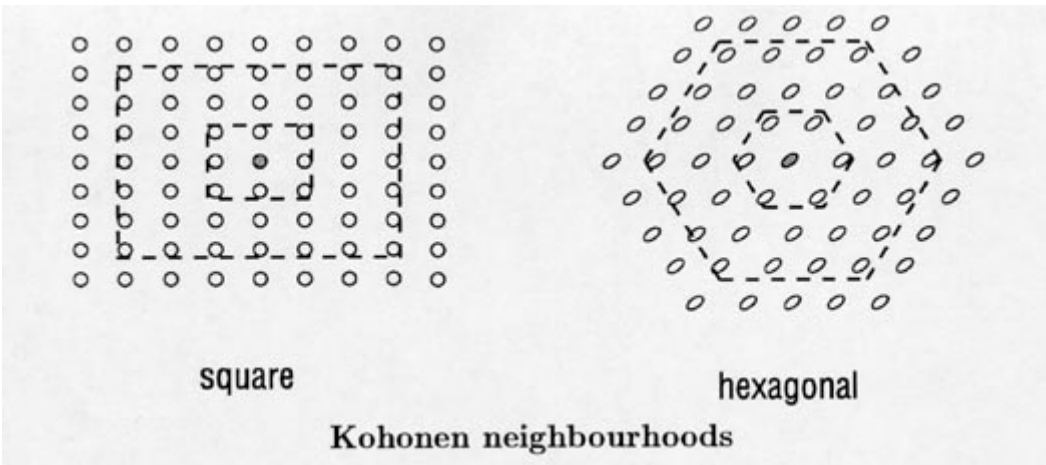
- 算法中，η(t)是可变学习速度，随时间的增加而减小。也就是说，随着训练过程的进行，权值的调整幅度越来越小
- $NE_{j^*}(t)$也随着时间而收缩，最后t足够大时，$NE_{j^*}(t)=\{N_{j^*}\}$，即只训练获胜结点本身
- η(t)和$NE_{j^*}(t)$有多种不同的形式，在具体训练过程中可以根据不同的要求不同的数据分布进行设计。

SOM Algorithm



Learning

- Decreasing the neighbor ensures progressively finer features are encoded
- gradual lowering of the learn rate ensures stability



Kohonen neighbourhoods

Basic Algorithm

- Initialize Map (randomly assign weights)
- Loop over training examples
  - Assign input unit values according to the values in the current example
  - Find the "winner", i.e. the output unit that most closely matches the input units, using some distance metric, e.g.

For all output units j=1 to m
and input units i=1 to n
Find the one that minimizes:

$$\sqrt{\sum_{i=1}^{n}\left(W_{ij}-I_i\right)^2}$$

  - Modify weights on the winner to more closely match the input

$$\Delta W^{t+1} = c(X_i^t - W^t)$$

where c is a small positive learning constant
that usually decreases as the learning proceeds

一种降维方法：随机映射

- 向量空间模型中，经常存在向量维数太高的问题，影响处理速度。因此，要在保持向量所表达的意义的基础上，给向量降维
- 设e_i表示这样的一个向量：除第i维的值为1外，其余处处为0
- n_{ji} 表示第i个项在第j篇文档中出现的次数（n_{ji}可以是其它各种形式的值）。n_j为第j篇文档的向量表示形式，则n_j的一种表示方式是：

$$n_j = \sum_k n_{jk} e_k$$

一种降维方法：随机映射

- 现在，用一个比e_k维数低的向量r_k代替e_k，r_k已被标准化为单位长度。代替后，第j篇文档表示为：$x_j = \sum_k n_{jk} r_k$
- 设矩阵R是由向量r_k构成的，R的第k列为r_k。根据上面两式，x_j可以用矩阵乘法来表示：

$$x_j = Rn_j$$

一种降维方法：随机映射

- 压缩必须以不改变向量对文档相似性度量的能力为前提。向量之间的相似性可以用它们的内积来度量： $x_j^T x_k = n_j^T R^T R n_k$
- R^TR可以分解为两部分：

其中，$\varepsilon_{ij} = r_i^T r_j \quad , i \neq j \quad , \quad \varepsilon_{ii} = 0 \qquad R^T R = I + \varepsilon$

如果 ε 矩阵的所有分量均为0，则有$R^TR=I$。这样，就保持了文档之间相似性。

一种运用统计方法选择r_k的方式是：r_k的各个分量相互独立，来自于期望为0的正态分布，将r_k的长度标准化为1。

自组织语义图

- 利用词与词在文档中的上下文关系，将词表示成一个向量，然后用表示词的向量作为SOM网络的输入，聚类，输出形成一个词汇类别图（Word category map）
- 在这个图中，意义相近的词聚在一起，组成一个词类，词在词汇类别图中的位置可以通过快速Hash的方法查到。

自组织语义图：一种将词向量化的方法

- 在一个文档集中考虑词与词之间的上下文关系。设$l_i^{(d)}$表示相对于第i个词位移为d的位置上出现的词集合（有可能出现多次），例如，$l_i^{(1)}$表示第i个词的所有前趋邻接词
- 用向量x_i表示第i个词，对位移集$\{d_1,\ldots,d_N\}$：

$$x_i = \left[x_i^{(d_i)},\ldots,x_i^{(d_N)}\right]^T, \qquad x_i^{(d)} = \frac{1}{\left|I_i^{(d)}\right|}\sum_{k\in I_i^{(d)}} e_k$$

其中，$\left|I_i^{(d)}\right|$表示$I_i^{(d)}$中词的数量

一般地，为计算简单，只取d=1和d=-1

自组织语义图：另一种对中文词汇向量化的方法

- 对每一个词，在文档集合中出现该词的时候会伴随一些修饰词。因此可以用修饰词来表示该词，提供该词的一些语义信息
- 例如对名词“大学”，会出现一些修饰词如“本科”、“重点”、“合格”等，则定义，大学={本科，重点，合格，…}

自组织语义图：另一种对中文词汇向量化的方法

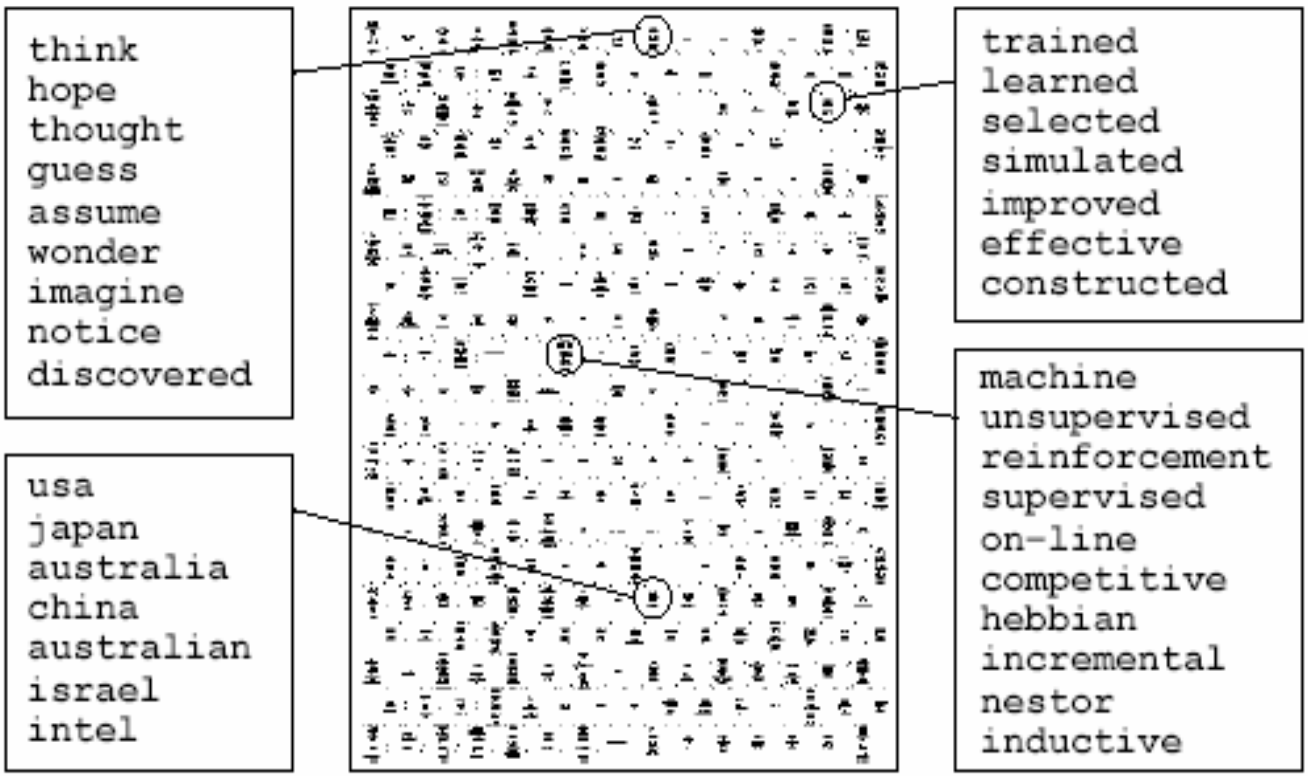
- 一般地，词w_i(i=1,2,…,N)为：
  其中n_i表示修饰词个数 $w_i = \{a_1^{(i)}, a_2^{(i)}, ..., a_{n_i}^{(i)}\}$
- 定义词w_i的向量表示为：
- 其中， $V(w_i) = [d_{i1}, d_{i2}, ..., d_{iN}]^T$

$$d_{ij} = \frac{(n_i - c_{ij}) + (n_j - c_{ij})}{n_i + n_j - c_{ij}}, \quad i \neq j \quad ; d_{ij} = 0, \quad i = j$$

c_{ij}表示词w_i和词w_j的修饰词集合中都出现的修饰词个数

- 将词用这种向量表示后，作为一个SOM网络的输入，可以聚类形成中文语义图。

自组织语义图：示例


think
hope
thought
guess
assume
wonder
imagine
notice
discovered
usa
japan
australia
china
australian
israel
intel
trained
learned
selected
simulated
improved
effective
constructed
machine
unsupervised
reinforcement
supervised
on-line
competitive
hebbian
incremental
nestor
inductive

利用SOM进行文本聚类：预处理

- 去掉非文本信息
- 去掉在整个文档集合中出现次数小于50次的词
- 去停用词
- 经过上述处理后，词的个数由 1 127 184 减少为 63 773

利用SOM进行文本聚类：Word category map

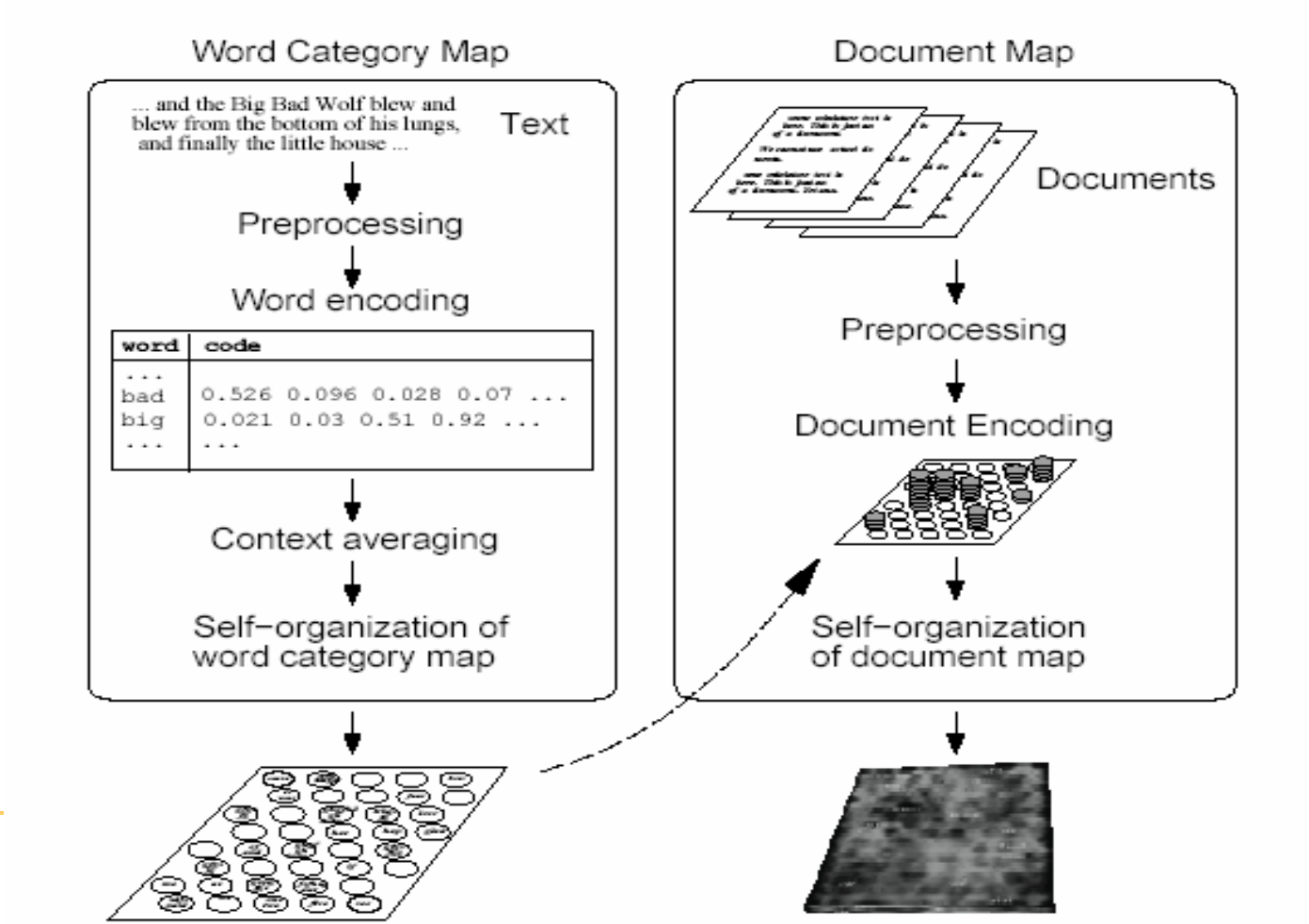
- 将词表示成为一个180维的向量，作为一个SOM网络的输入，进行聚类
- 最终产生 13 432 个词类单元

（ 63 773 → 13 432）

利用SOM进行文本聚类：Document map

- 利用上面产生的词类将文档向量化后，每篇文档表示为一个 13 432 维的向量，再利用随机映射(Random mapping method)的降维方法，向量维数减少到 315 维
- 将这 315 维的向量作为一个 SOM 的输入
- 相关的结果可以参见 http://websom.hut.fi/websom/

利用SOM进行文本聚类

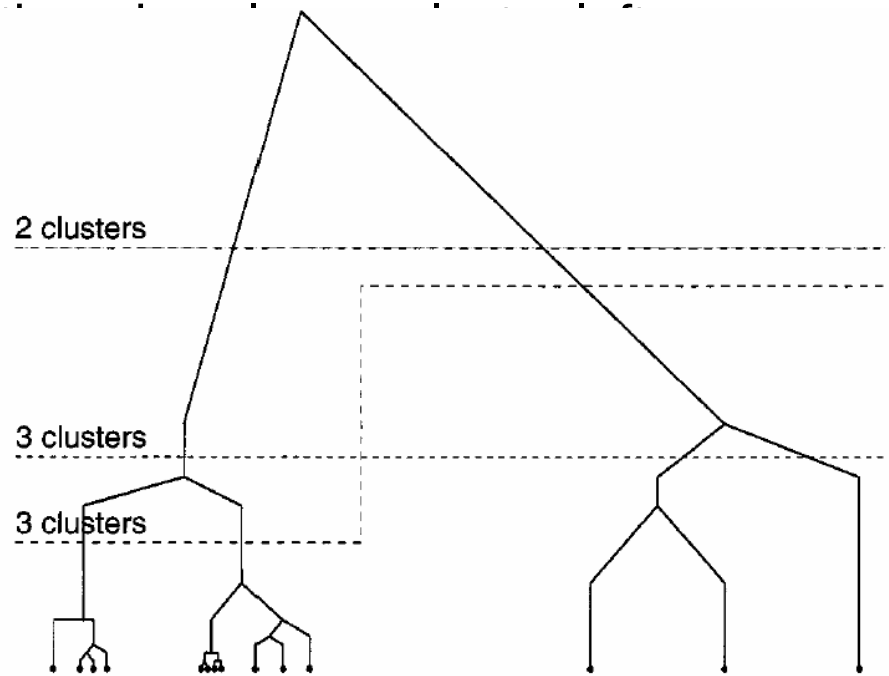


Hierarchical and Partitive Approaches

- Partitive algorithm
  - Determine the number of clusters.
  - Initialize the cluster centers.
  - Compute partitioning for data.
  - Compute (update) cluster centers.
  - If the partitioning is unchanged (or the algorithm has converged), stop; otherwise, return to step 3
- k-means error function
  - To minimize error function $E = \sum_{k=1}^{C} \sum_{x \in Q_k} \|x - c_k\|^2$

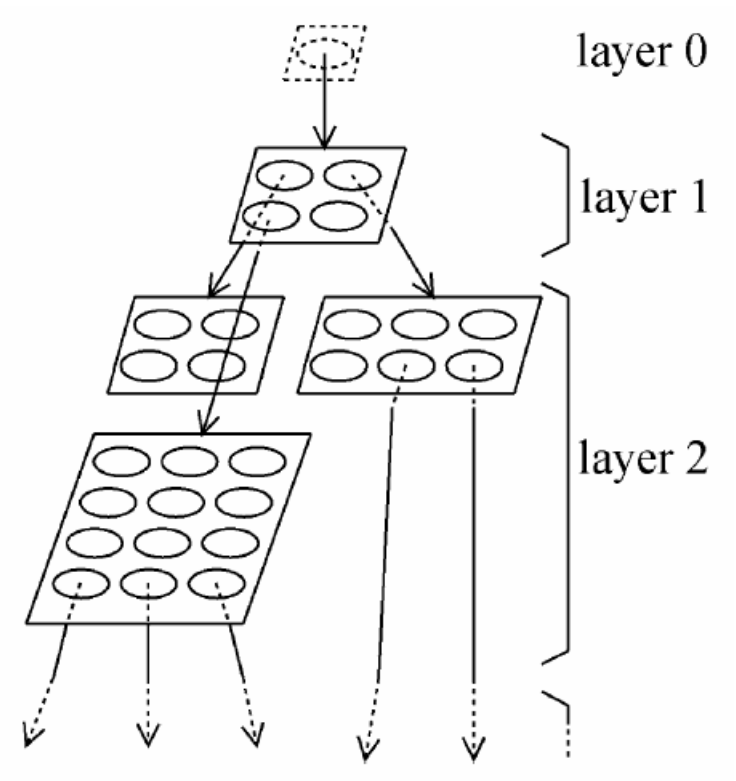
Hierarchical and Partitive Approaches

- Hierarchical clustering algorithm (Dendrogram)
  - Initialize: Assign each vector to its own cluster
  - Compute distances between all clusters.
  - Merge the two clusters that are closest to each other.
  - Return to step 2 until
- Partition strategy
  - Cut at different level



Hierarchical SOM

- GHSOM – Growing Hierarchical Self-Organizing Map
  - grow in size in order to represent a collection of data at a particular level of detail



神经网络集成

1996年，Sollich和Krogh 将神经网络集成定义为：“神经网络集成是用有限个神经网络对同一个问题进行学习，集成在某输入示例下的输出由构成集成的各神经网络在该示例下的输出共同决定”。

神经元整体

目前人工神经网络的研究不具有从信息处理的整体结构进行系统分析的能力，因此，很难反映出人脑认知的结构。 由于忽视对于整体结构和全局结构的研究，神经网络对于复杂模型组织结构的层次化和功能模块化组织机理还处于十分无知的情况。

功能柱

20世纪60年代末，美国科学家发现，在大脑视觉皮层中，具有相同图像特征选择性和相同感受野位置的众多神经细胞，以垂直于大脑表面的方式排列成柱状结构————功能柱。30多年来，脑研究领域一直将垂直的柱状结构看作大脑功能组织的一个基本原则。但是，传统的功能柱研究还不能阐释视觉系统究竟是如何处理大范围复杂图像信息的。

功能柱

中科院上海生命科学研究院神经科学研究所李朝义实验室通过对猫的视皮层的研究，发现在初级视皮层中存在一种与处理大范围复杂图形特征有关的功能结构。与目前所有已知结构不同，它不是柱状的，而是形成许许多多直径约300微米的小球，分散地镶嵌在已知的垂直功能柱中。这是在简单特征功能柱基础上所形成的第二级功能筑构，处理各种更复杂的图像信息。视觉系统可能正是通过这种神经机制，以有限的信息量把目标物从复杂的背景图像中分离出来。

功能柱

- 1972年：Wilson-Cowan方程来描述功能柱；
- 1990年：Shuster等人模拟视皮层中发现的同步振荡；
- 1993年：Jansen等人提出了耦合功能柱模型产生了类EEG波形和诱发电位；
- 1994年：Fukai设计了功能柱式的网络模型来模拟视觉图样的获取；
- 1997年：Hansel等人根据视皮层朝向柱的结构构建了一个超柱模型，研究其中的同步性和混沌特性，并对朝向选择性的功能柱机理做出解释；
- 1998年：Fransén等人把传统网络中的单细胞代换成多细胞构成的功能柱，来模拟工作记忆

功能柱

中国科学院生物物理所 视觉信息加工重点实验室
中国科学院计算研究所
李速① 齐翔林① 胡宏② 汪云九①

功能柱结构神经网络模型中的同步振荡现象

功能柱是一个振荡子，而且表明功能柱可以成为皮层多样化的节律活动的发生源，EEG中的各种节律均可以在结构具有普遍性的功能柱中找到生理基础。

功能柱

Rose-Hindmarsh方程来描述单神经元:

$$\dot{x} = y + ax^3 - bx^2 - z + I_{syn} + I_{stim}$$

$$\dot{y} = c - dx^2 - y$$

$$\dot{z} = r\left[s\left(x - x_0\right) - z\right]$$

x: 代表膜电位,
y: 表示快速回复电流,
z: 描述慢变化的调整电流,
I_{syn} 表示突触电流,
I_{stim} 表示外界输入

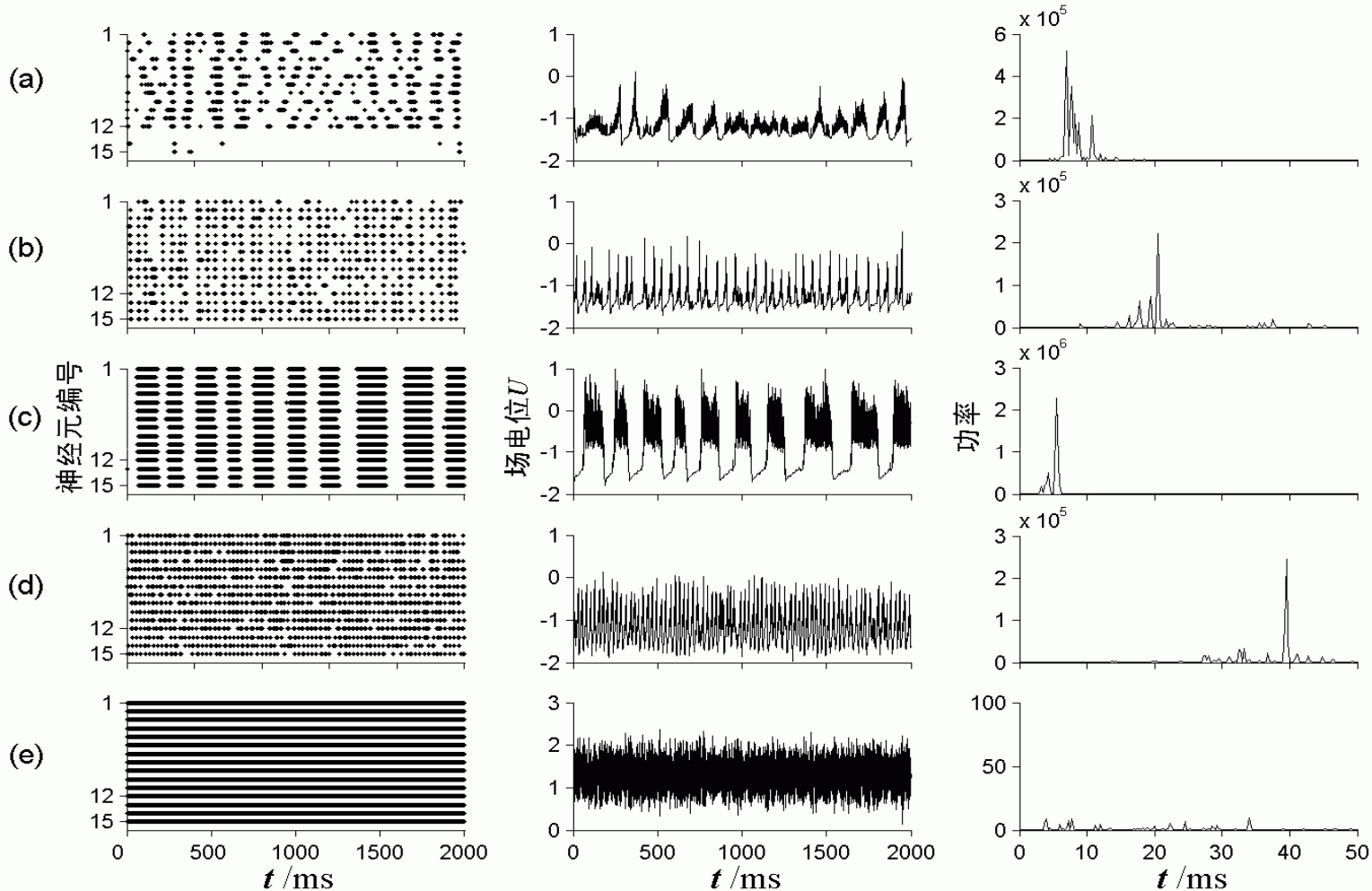
功能柱

模型采用基于电流的突触模型，在突触前细胞的每个动作电位都将触发突触后细胞的I_{syn}输入。突触电流I_{syn}表示为:

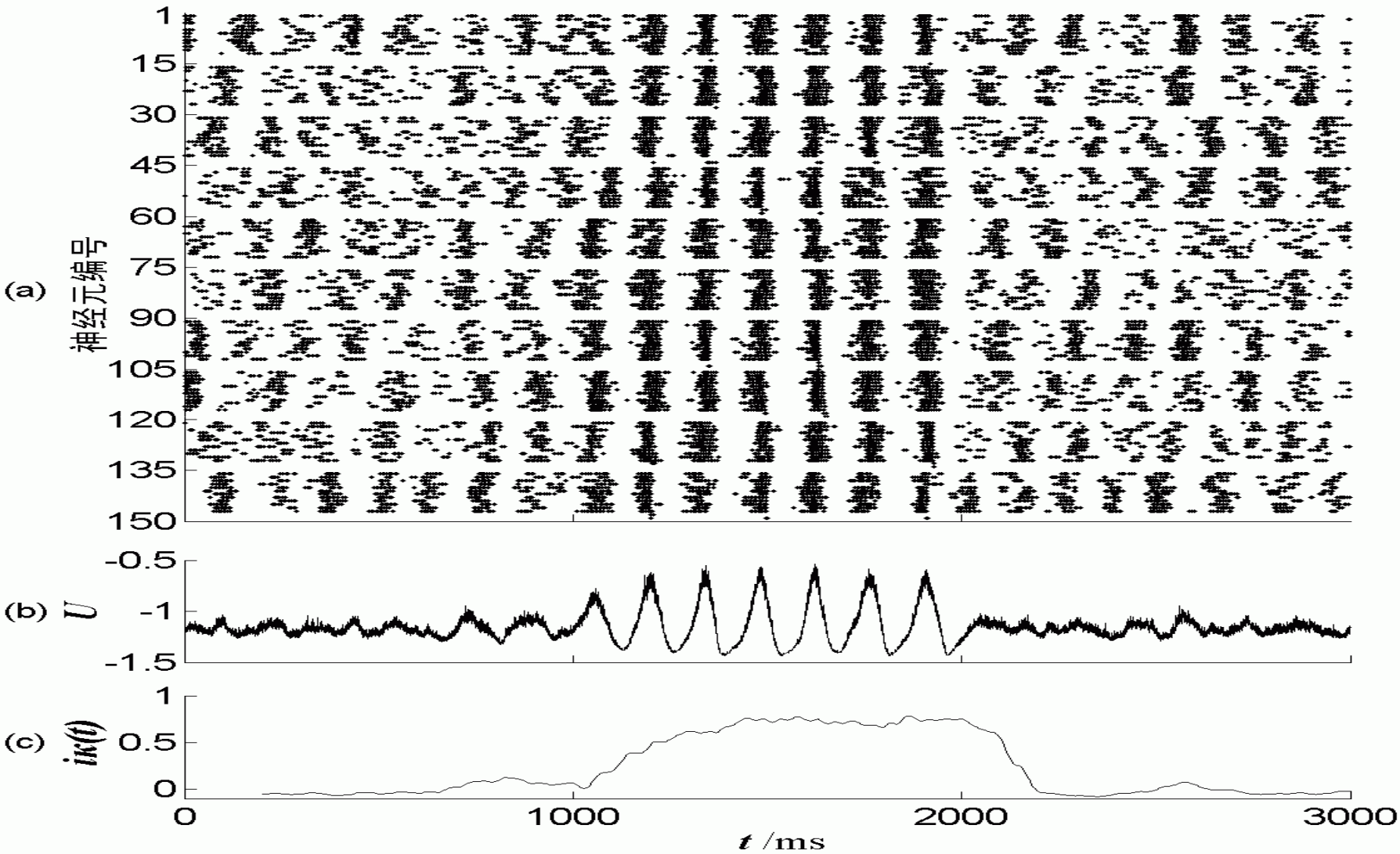
$$I_{syn} = g_{syn} V_{syn} (e^{-t/\tau_1} - e^{-t/\tau_2})$$

g_{syn} 为膜电导
τ_1, τ_2 时间常数
V_{syn} 表示突触后电位

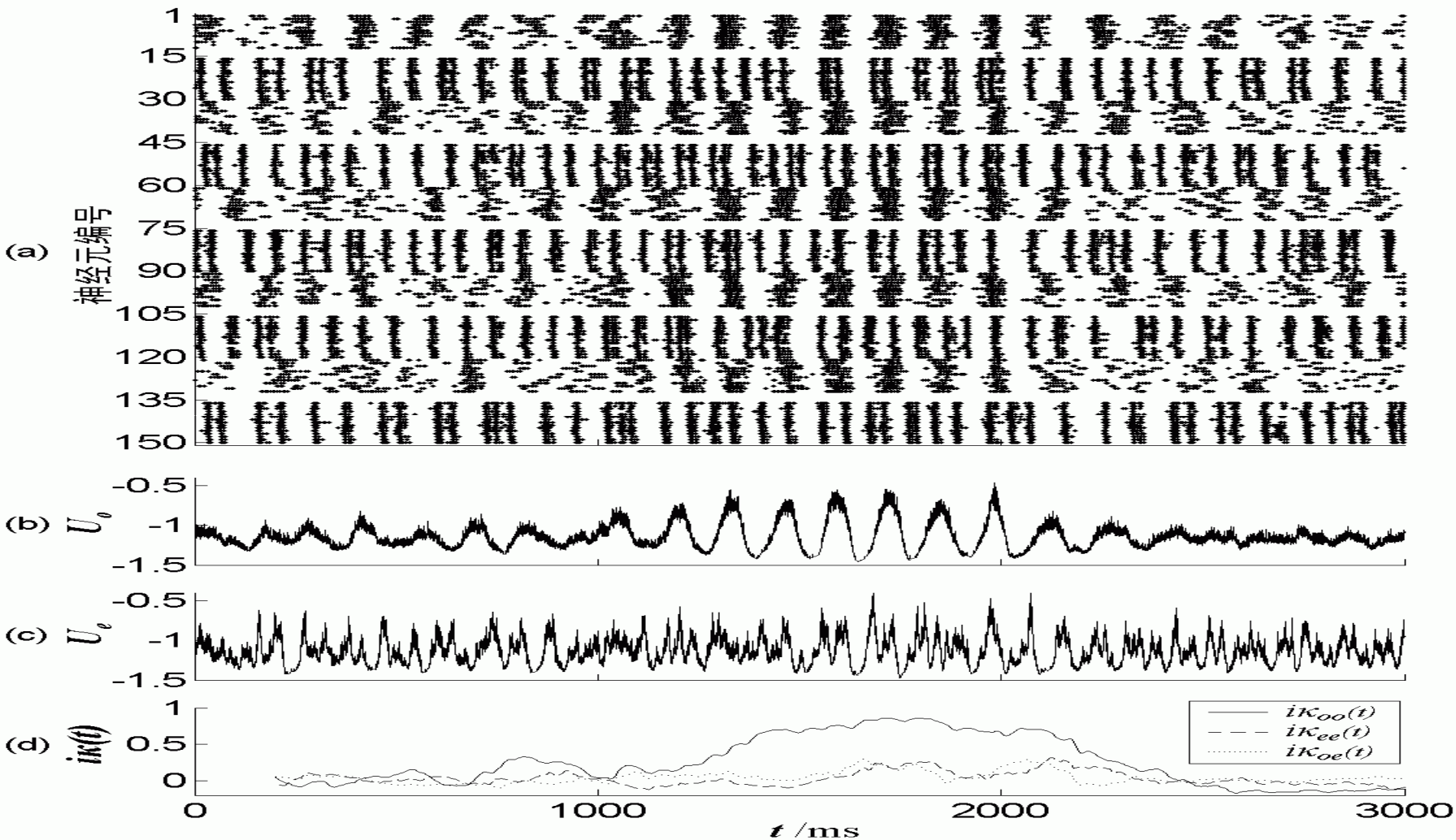
功能柱


(a)
(b)
(c)
(d)
(e)
神经元编号
场电位U
功率
x 10^5
x 10^6
t /ms
0
500
1000
1500
2000
10
20
30
40
50

同源功能柱


(a)
神经元编号
1
15
30
45
60
75
90
105
120
135
150
(b)
U
-0.5
-1
-1.5
(c)
ik(t)
1
0.5
0
0
1000
2000
3000
t /ms

异源功能柱


(a)
神经元编号
1
15
30
45
60
75
90
105
120
135
150
(b)
U_o
-0.5
-1
-1.5
(c)
U_e
-0.5
-1
-1.5
(d)
$ik(t)$
1
0.5
0
$ik_{oo}(t)$
$ik_{ee}(t)$
$ik_{oe}(t)$
0
1000
2000
3000
t /ms

功能柱

功能柱是一个振荡子，而且表明功能柱可以成为皮层多样化的节律活动的发生源，EEG中的各种节律均可以在结构具有普遍性的功能柱中找到生理基础。

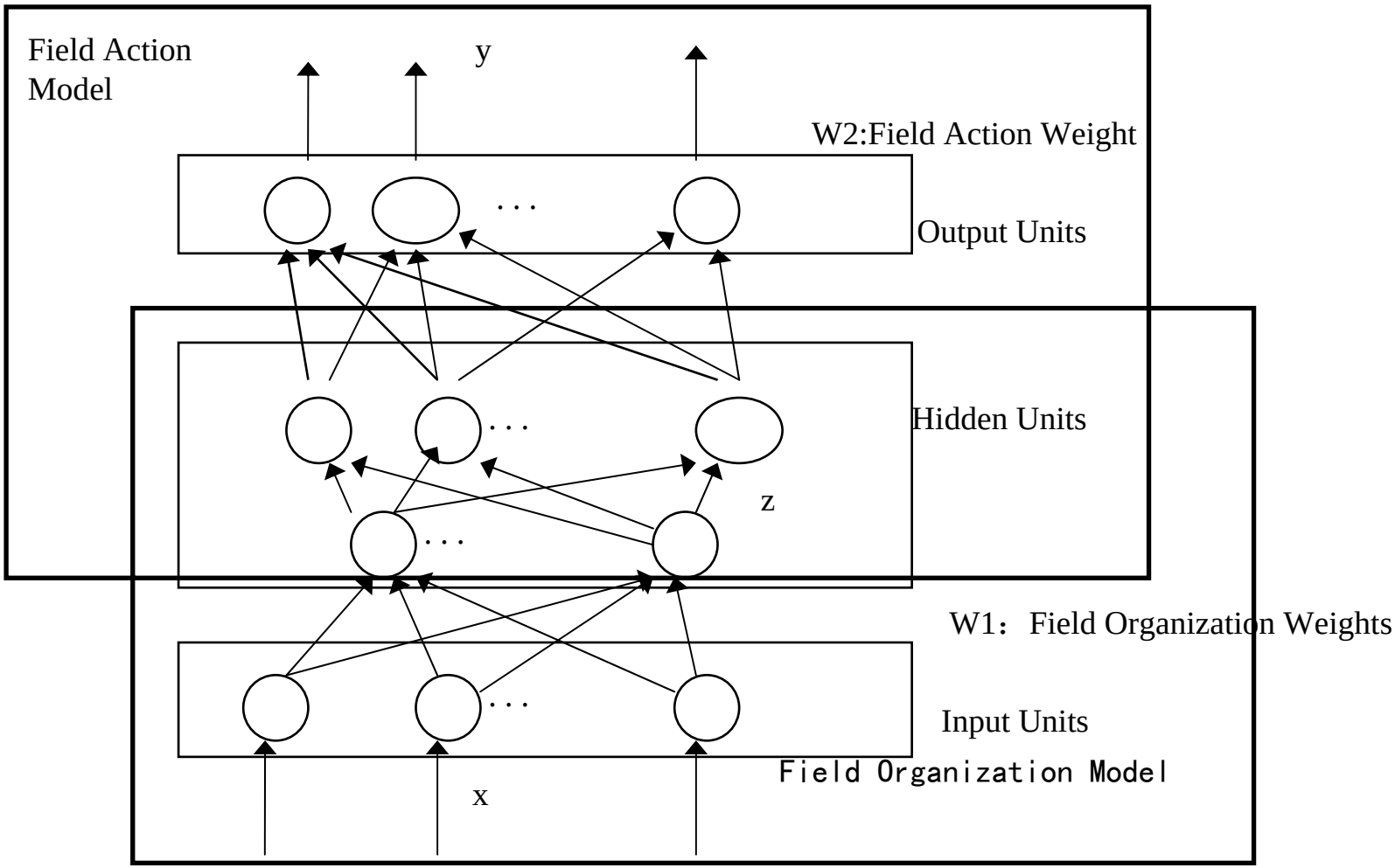
功能柱

功能柱是介于单神经元和皮层脑区之间的一种中间层次的单元，理解这种中间层次的单元的活动特点，能够为脑科学中微观现象和宏观现象的研究之间建立一座桥梁

神经场

神经场研究的出发点是信息处理系统的整体结构，一般的系统表示为非欧氏空间（在一定拓扑结构下形成流形）。研究的一个关键就是建立环境结构流形与神经流形的耦合关系，用流形的思想、拓扑的概念和统计推理来研究整体结构所具有的性质，利用整体不变性质，处理和分析表示结构与神经流形的优化逼近过程。

神经场


Field Action
Model
y
W2:Field Action Weight
Output Units
Hidden Units
z
W1：Field Organization Weights
Input Units
Field Organization Model
x

神经场

神经场理论框架，体现整体信息处理的结构在两方面：一方面，表示结构的编码和模型结构通过拓扑结构进行表示，具有层次化、模块化的组织，形成树型链结构，模型结构具有扩展成无限模型的性质和分维组织机理，分解成层次化的结构。我们试图用代数拓扑的方法来描述这种结构，体现整体不变性质，神经网络具有对于结构进行模型化的机理，对于系统结构的逼近。另一方面，复杂的模型由简单的模型集成，简单的模型嵌入更复杂的结构中，信息几何研究局部和整体不变度量的关系，研究全局不变量，即研究学习的全局优化过程。

脑功能区之间的耦合

对于以细胞、分子事件为基础的局部神经网络如何组装起来构成庞大的复杂的脑来实现高级功能，既缺少有成效的研究手段，在理论上也只有很模糊的想法。

展望

我们要创立一系列新方法，包括若干原理上全新的方法，把离子通道、突触、神经元的工作机理与脑的高级功能沟通起来。

基于神经生物学的最新研究成果，开展神经计算的研究。

References

[1] Simon Haykin. Neural Networks: A Comprehensive Foundation. 2nd Edition, 1998, Prentice Hall，叶世伟，史忠植译

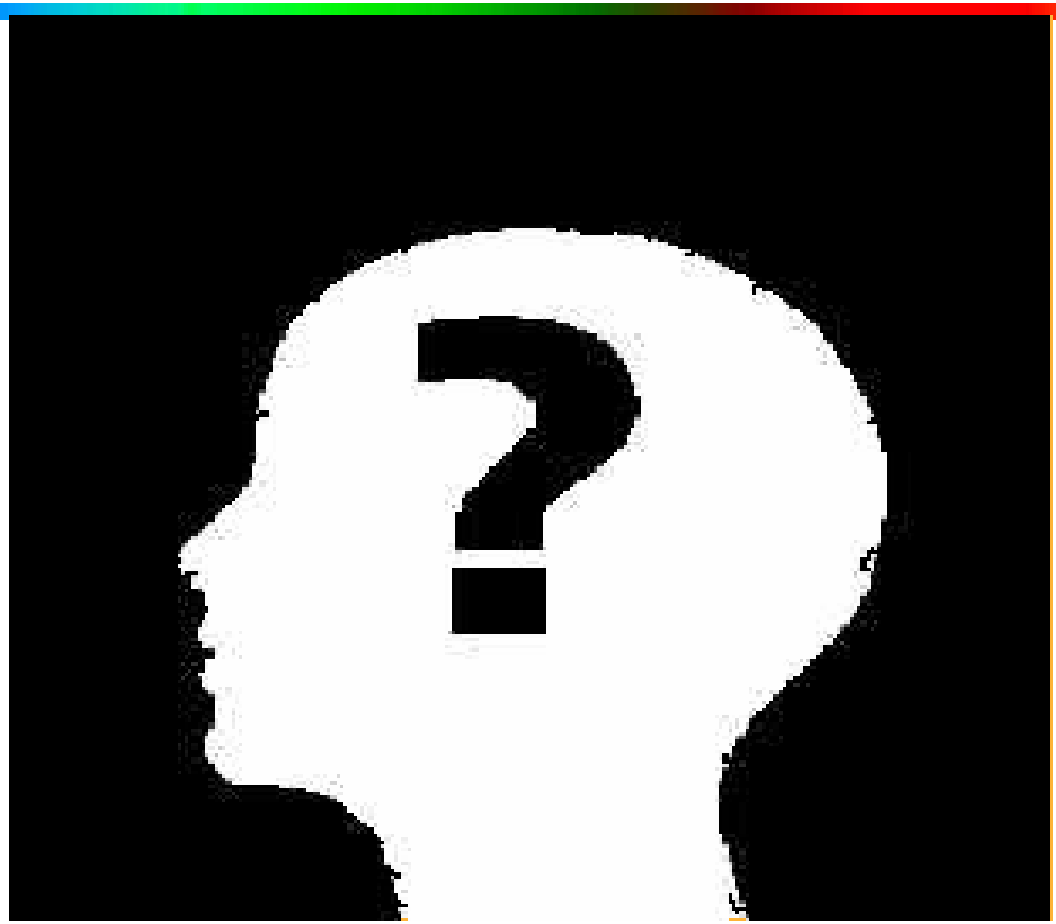
[2] Neural Networks Software, URL: http://cortex.snowseed.com/cortex.htm

[3] Sun Hwa Hahn. Neural Network (IV): Hopfield Net & Kohonen's Self-Organization Net. Information & Communication University. http://ai.kaist.ac.kr/~jkim/cs570-2000/Lectures/Dr.SHHahn's/NeuralNetwork4.**ppt** [

[4] Harry R. Erwin. Neural Network Toolbox. http://www.cet.sunderland.ac.uk/

[5] 史忠植. 神经计算. 电子工业出版社, 1994

www.intsci.ac.cn/shizz



Thank you !!!