第四章 计算智能

主要内容：

§ 4.1 人工神经网络

§ 4.2 模糊计算

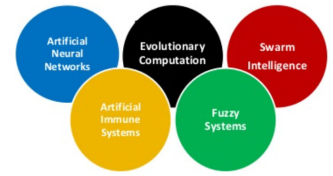
§ 4.3 进化算法与遗传算法

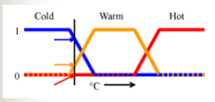
§ 4.4 粒群优化算法

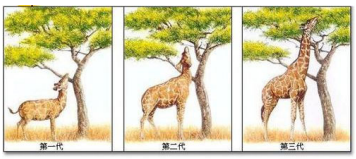
§ 4.5 蚁群算法

计算智能及分类

- 计算智能是信息科学与生命科学相互交叉的前沿领域。
- 典型方法
  - 模糊逻辑
  - 神经计算
  - 进化计算
  - 群智能



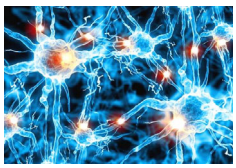




长颈鹿的进化示意图

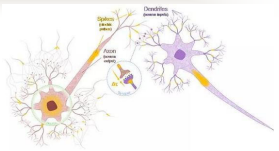
AI vs. CI

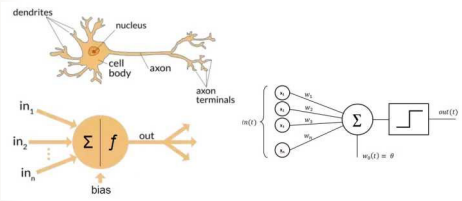
- The huge majority of AI/CI researchers concerned with the subject sees them as different areas
  - CI forms an alternative to AI
    - Seeking similar goals
    - CI is buried deeply within the core of the system
    - AI is at the outer level
  - AI subsumes CI



§ 4.1 人工神经网络

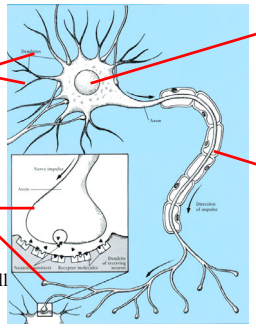
- 生物神经网络模型
- 人工神经元模型
- 人工神经网络原理
- 人工神经网络结构和学习





神经元生理构造

1. Biological Neurons (生物神经元)



Dendrites (树突)
carry electrical *into* the cell body

Cell Body (细胞体)
5–10 microns in diameter, sums and thresholds these incoming signals

Synapse (突触)
contact between an axon of one cell and a dendrites of another cell

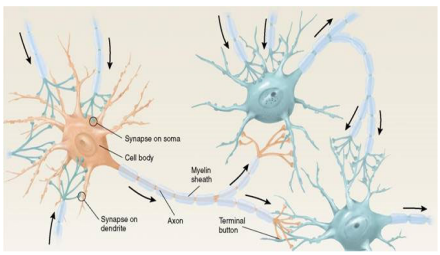
Axon (轴突)
carry the signal from the cell body *out* to other neurons

A Neuron (单神经元)：多输入/一输出单元

神经元连接

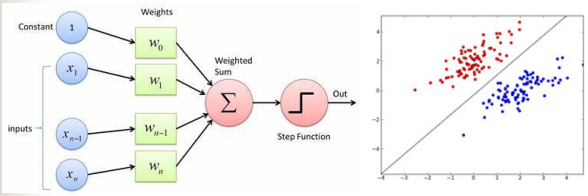
- 学习方法

通过突触的大小和强度随经验而变化来实现.



什么是感知器

It's an artificial neuron using *step function* for activation to produces binary output, usually used to classify the data into two parts, therefore, it is also known as a *Linear Binary Classifier*.

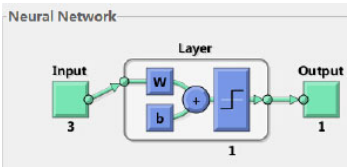


什么是感知器

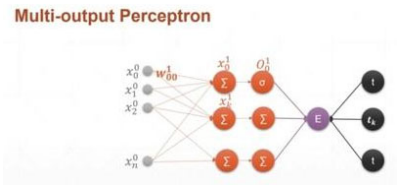
- Perceptron (Rosenblatt 1958)
  - Three layer system:
    - Input nodes, output node, association layer
  - Can learn to connect or associate a given input to a random output unit
- Minsky and Papert
  - Showed that a single layer perceptron cannot learn the *XOR* of two binary inputs
  - Lead to loss of interest (and funding) in the field

单层感知器分类

- 单层单输出感知器



- 单层多输出感知器



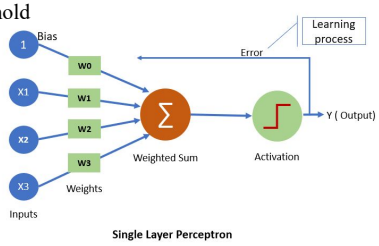
单层多输入单输出感知器

Input: x1, x2, x3…

Weights: w1, w2, w3…

Output: 0 or 1 based on the weighted sum is less than or greater than threshold

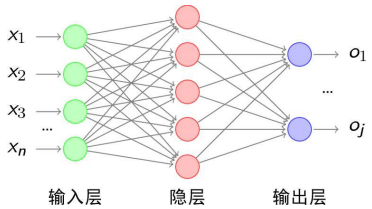
Bias: 1



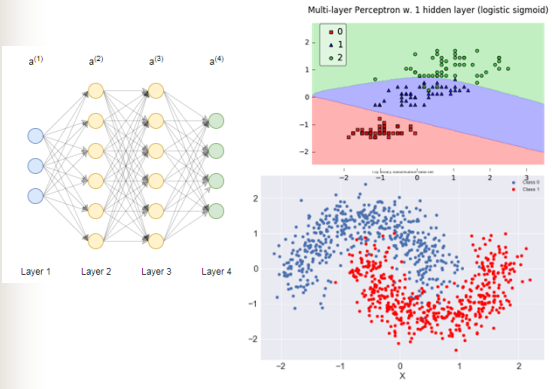
Single Layer Perceptron

多层感知器

Multi-Layer Perceptron Neural Network has an input layer, hidden layer and output layer. Input layer feed the input data set is came from feature extraction and output layer produced the set of output vector.

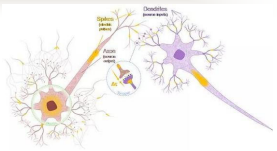


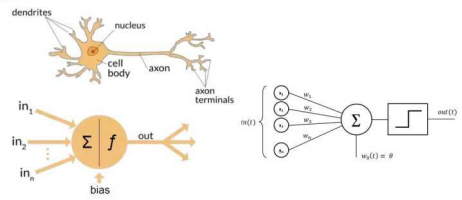
多层感知器用作分类器



§ 4.1 人工神经网络

- 生物神经网络模型
- 人工神经元模型
- 人工神经网络原理
- 人工神经网络结构和学习

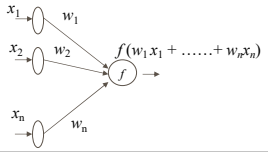




Artificial neuron model

- Inputs to the network are represented by the mathematical symbol, x_n
- Each of these inputs are multiplied by a connection weight, w_n

  $\text{sum} = w_1 x_1 + \ldots\ldots + w_n x_n$
- These products are simply summed, fed through the transfer function $f(\cdot)$ to generate a result and then output.



- 符号说明

➢ 标量: 斜体小写字母

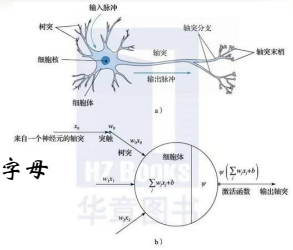
E.g., *a*, *b*, *c*

➢ 向量: 加粗非斜体小写字母

E.g., **a**, **b**, **c**

➢ 矩阵: 大写加粗非斜体字母

E.g., A, B, C



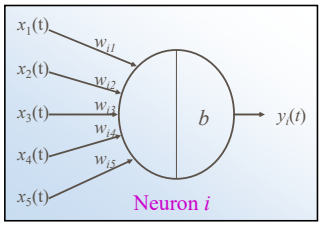
1. 人工神经元模型

(1) 刺激

$$u_i(t) = \sum_j w_{ij} \cdot x_j(t)$$

(2) 响应

$$y_i(t) = f(b + u_i(t))$$



b = *threshold* or *bias*

$y_i(t)$ = output of neuron *i* at time *t*

w_{ij} = connection strength between neuron *i* and neuron *j*, *weight*

f = *transfer function*, or *activation function*

(3) 阈值与权重

可将阈值b看作权重，则其输入为常量1

$$\begin{cases} u_i = \sum_{j=1}^{n} w_{ij} x_j \\ y_i = f(u_i + b) \end{cases} \Rightarrow \begin{cases} u_i = b + \sum_{j=1}^{n} w_{ij} x_j = \sum_{j=0}^{n} w_{ij} x_j \\ y_i = f(u_i) \end{cases}$$

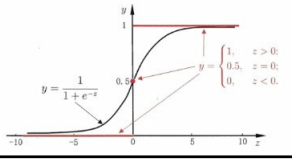
其中，$b = x_0, w_{i0} = 1$

- 阈值b可省略
- 阈值b和权重w是可调节的标量
- 根据学习规则调节它们，使神经元输入输出满足目标

(4) 传递函数

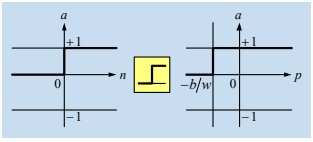
- 传递函数可以是网络输入的线性或非线性函数
- 三个最常用的函数
  - Hard limit 传递函数
  - Linear 传递函数
  - Log-sigmoid 传递函数

$$y = \begin{cases} 0, & z < 0; \\ 0.5, & z = 0; \\ 1, & z > 0, \end{cases}$$



Hard Limit

Hard limit 传递函数



a=*hardlim*(*n*)　　*a*=*hardlim*(*wp*+*b*)

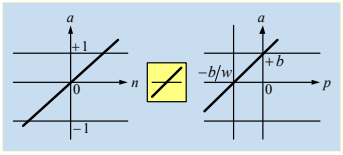
$a = 0, \quad \text{if } n < 0$

$a = 1, \quad \text{if } n \geq 0$

MATLAB function: *hardlim*

Linear

Linear 传递函数



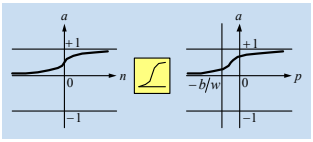
a=*purelin*(*n*)　　*a*=*purelin*(*wp*+*b*)

$a = n$

MATLAB function: *purelin*

Log-sigmoid

Log-sigmoid 传递函数



a=*logsig*(*n*)　　*a*=*logsig*(*wp*+*b*)

$a = 1/[1+exp(-n)]$

MATLAB function: *logsig*

线性阈值单元

2. 线性阈值单元(TLU)

TLU的基本形式

$$y = f(\sum_{j=1}^{d} w_j x_j + w_0) = f(\mathbf{w}^T\mathbf{x}) = \begin{cases} 0, & \mathbf{w}^T\mathbf{x} < 0 \\ 1, & \mathbf{w}^T\mathbf{x} >= 0 \end{cases}$$

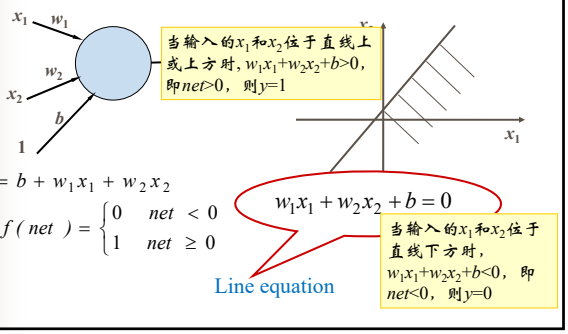
$$\mathbf{w} = [w_0, w_1, ..., w_d]^T$$

$$\mathbf{x} = [1, x_1, ..., x_d]^T$$

TLU的基本功能—线性划分

线性阈值单元

当有两个输入x_1和x_2时，若将x_1和x_2分别看成平面上的横轴和纵轴，则不同值将对应该平面上的不同点。



$net = b + w_1x_1 + w_2x_2$

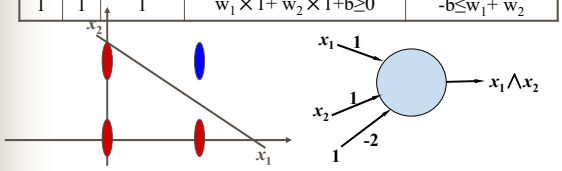
$$y = f(net) = \begin{cases} 0 & net < 0 \\ 1 & net \geq 0 \end{cases}$$

线性阈值单元

- 对于两个输入，TLU通过权值阈值决定的直线，将平面上的点划分为两类，分别对应神经元的兴奋和抑制状态；
- 对于三个输入，TLU通过权值阈值决定的平面，将空间中的点划分为两类；
- 对于多个输入来说，权值阈值对应的就是一个超平面，将超空间中的点进行分类；
- 单个TLU解决的问题为线性可分的。

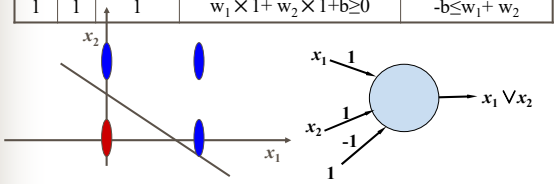
TLU 实现 AND

Input		Output	Function	Condition
x_1	x_2	$x_1 \wedge x_2$	$w_1 \times x_1 + w_2 \times x_2 + b = 0$	
0	0	0	$w_1 \times 0 + w_2 \times 0 + b < 0$	$b < 0$
0	1	0	$w_1 \times 0 + w_2 \times 1 + b < 0$	$-b > w_2$
1	0	0	$w_1 \times 1 + w_2 \times 0 + b < 0$	$-b > w_1$
1	1	1	$w_1 \times 1 + w_2 \times 1 + b \geq 0$	$-b \leq w_1 + w_2$



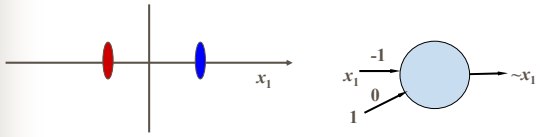
TLU 实现 OR

Input		Output	Function	Condition
x_1	x_2	$x_1 \vee x_2$	$w_1 \times x_1 + w_2 \times x_2 + b = 0$	
0	0	0	$w_1 \times 0 + w_2 \times 0 + b < 0$	$b < 0$
0	1	1	$w_1 \times 0 + w_2 \times 1 + b \geq 0$	$-b < w_2$
1	0	1	$w_1 \times 1 + w_2 \times 0 + b \geq 0$	$-b < w_1$
1	1	1	$w_1 \times 1 + w_2 \times 1 + b \geq 0$	$-b \leq w_1 + w_2$



TLU 实现 NOT

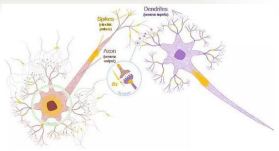
Input	Output	Function	Condition
x_1	$\sim x_1$	$w_1 \times x_1 + b = 0$	
0	1	$w_1 \times 0 + b \geq 0$	$b \geq 0$
1	0	$w_1 \times 1 + b < 0$	$-b > w_1$

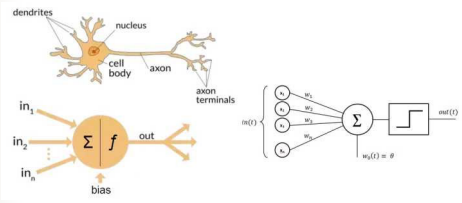


NOT: Let threshold be 0, single input with a negative weight

§ 4.1 人工神经网络

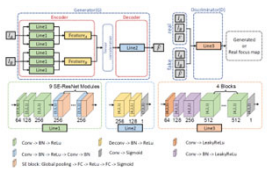
- 生物神经网络模型
- 人工神经元模型
- 人工神经网络原理
- 人工神经网络结构和学习





Artificial neural networks

- The structure of an artificial neuron, the basic component of artificial neural networks (source: Wikipedia).
- One of the most influential technologies of the past decade is *artificial neural networks*, the fundamental piece of *deep learning* algorithms, the bleeding edge of artificial intelligence.



Artificial neural networks

- *Artificial Neural Network (ANN)* are programs designed to solve any problem by trying to mimic the structure and the function of our nervous system.
- Neural networks are based on simulated neurons, which are joined together in a variety of ways to form networks.
- Neural network resembles the human brain in the following two ways:
  - A neural network acquires knowledge through learning.
  - A neural network's knowledge is stored within the interconnection strengths known as synaptic weight.

Features

- Features of artificial neural networks
  - The output values can be represented as a discrete value, a real value, or a vector of values
  - Tolerant to noise in input data
  - Time factor
    - It takes long time for training
    - Once trained, an ANN produces output values (predictions) fast
  - It is hard for human to interpret the process of prediction by ANN

Introduction

CONTD...

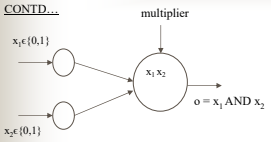


Fig 1: AND gate graph

This graph cannot be considered a neural network since the connections between the nodes are fixed and appear to play no other role than carrying the inputs to the node that computed their conjunction.

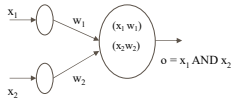


Fig 2: AND gate network

The graph structure which connects the weights modifiable using a learning algorithm, qualifies the computing system to be called an artificial neural networks.

• The field of neural network was pioneered by *BERNARD WIDROW* of Stanford University in 1950's.

人工神经网络

- Artificial neural networks

Artificial neural networks are composed of *an input layer*, which receives data from outside sources (data files, images, hardware sensors, microphone), *one or more hidden layers* that process the data, and *an output layer* that provides one or more data points based on the function of the network.

人工神经网络

- Artificial neural network

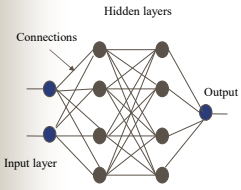


Fig. artificial neural network model

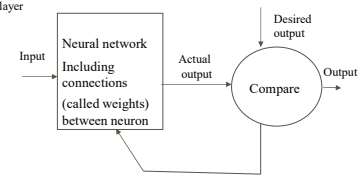
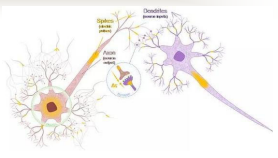
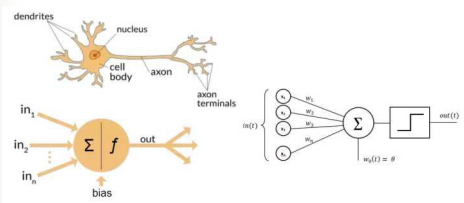


Figure showing adjust of neural network

§ 4.1 人工神经网络结构

- 生物神经网络模型
- 人工神经元模型
- 人工神经网络
- 人工神经网络结构和学习





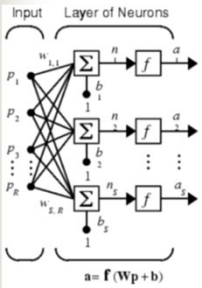
神经网络结构

(1) Neural network architectures

- Models are specified by the *three basic entities*
  - models of the *processing elements* (neurons) (神经元模型)
  - models of *inter-connections and structures* (network topology) (网络拓扑模型)
  - *learning rules* (the ways information is stored in the network)
- The *weights* may be *positive* (excitatory) or *negative* (inhibitory)
- *Information* is stored in the *connection weights*

单层神经网络

- Single layer neural network



- R: number of input
- S: number of neuron (node) in a layer ($R \neq S$)
- Input vector **p** is a vector of length R
- Bias vector **b** and output vector **a** are vectors of length S
- Weight matrix **W** is an $S \times R$ matrix

$$W = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1R} \\ w_{21} & w_{22} & \cdots & w_{2R} \\ \vdots & \vdots & \ddots & \vdots \\ w_{S1} & w_{S2} & \cdots & w_{SR} \end{bmatrix}$$

神经网络结构

- Other neural network architectures

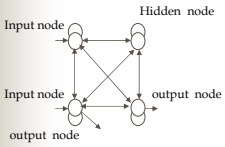


Fig: fully connected network

The neural network in which every node is connected to every other nodes, and these connections may be either excitatory (positive weights), inhibitory (negative weights), or irrelevant (almost zero weights).

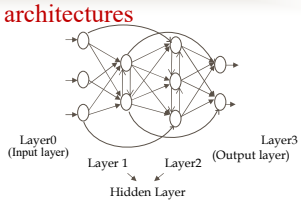


Fig: layered network

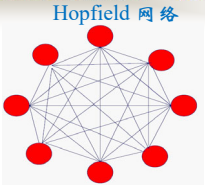
These are networks in which nodes are partitioned into subsets called layers, with no connections from layer j to k if $j > k$.

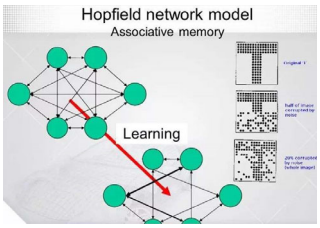
Hopfield网络

- Hopfield网络

Hopfield

Hopfield 网络





神经网络结构

- Other neural network architectures

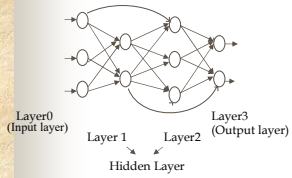


Fig : Acyclic network

This is the subclass of the layered networks in which there is no intra-layer connections. A connection may exist between any node in layer i and any node in layer j for $i < j$, but a connection is not allowed for $i = j$.

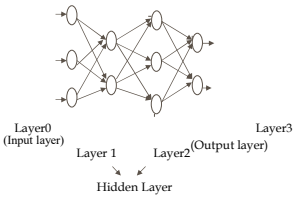


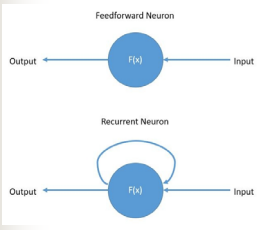
Fig : Feedforward network

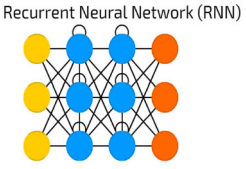
This is a subclass of acyclic networks in which a connection is allowed from a node in layer i only to nodes in layer $i+1$

网络拓扑结构

(2) Network topology architecture

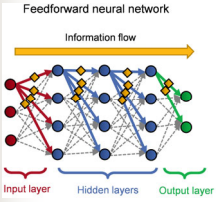
- Feedforward network (前馈网络)
- Recurrent network (递归网络)

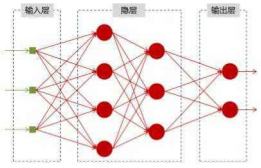




前馈网络

- Feedforward network (前馈网络)
  - When no node output is an input to a node in the same layer or preceding layer
  - It allows signals to travel one way only, from input to output. The output of any layer does not affect that same layer.

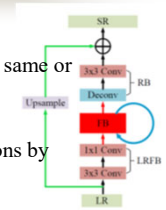


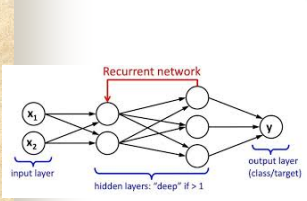


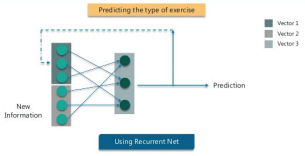
前馈全连接神经网络

反馈网络

- Recurrent network (递归网络)
  - When outputs are directed back as inputs to same or preceding-layer nodes.
  - It can have signals traveling in both directions by introducing loops in the network.







网络拓扑架构

- 如何选择网络拓扑架构

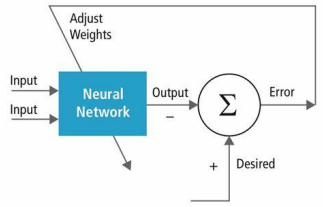
Problem specifications help define the network in the following ways:

- Number of network inputs = number of problem inputs
- Number of neurons in output layer = number of problem outputs
- Output layer transfer function choice at least partly determined by problem specification of the outputs.

ANN学习

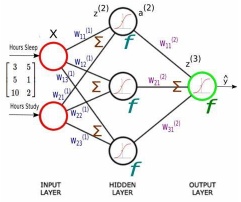
(3) Learning of ANN

- In artificial neural networks, learning refers to the method of modifying the *weights* of connections between the nodes of a specified network.
- The learning ability of a neural network is determined by its *architecture* and by the *algorithmic method* chosen for training.



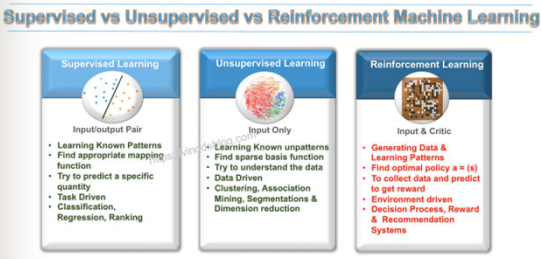
ANN学习

- Two kinds of learning in neural networks
  - *Parameter learning*, which concerns the updating the *weights* and the *bias* in a neural network
  - *Structure learning*, which focus on the change in the *network structure*, including the *number of nodes* and their *connection types*.



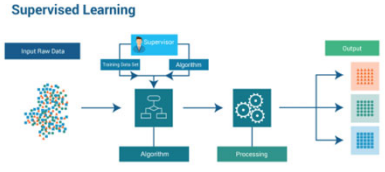
ANN Learning

- Each kind of learning can be classified into three categories
  - Supervised learning
  - Unsupervised learning
  - Reinforcement learning



有监督学习

- Learning rule：Supervised learning
  - A teacher is available to indicate whether a system is performing correctly, or to indicate the amount of error in system performance. Here a teacher is a set of training data.
  - The training data consist of pairs of input and desired output values that are traditionally represented in data vectors.
  - Supervised learning can also be referred as classification.



有监督学习

- Learning rule：Supervised learning
  - A set of examples (the *training set*)

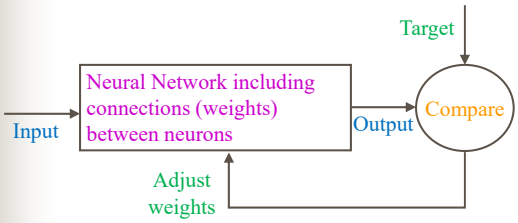
$$\{\mathbf{p}_1,\mathbf{t}_1\},\ \{\mathbf{p}_2,\mathbf{t}_2\},\ldots,\ \{\mathbf{p}_q,\mathbf{t}_q\}$$

where p_i is an input to the network and t_q is the corresponding correct (target) output.

  - As the inputs are applied to the network, the network outputs are compared to the targets.
  - Adjust the weights and biases of the network in order to move the *network outputs* closer to the *targets*.

有监督学习

Learning Rule：Supervised Learning



无监督学习

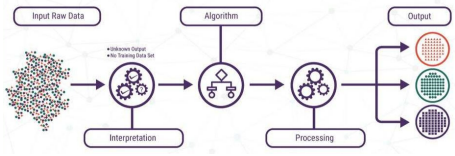
Learning Rule — Unsupervised Learning

- This is learning by doing.
- No sample outputs are provided to the network against which it can measure its predictive performance for a given vector of inputs.
- One common form of unsupervised learning is clustering where we try to categorize data in different clusters by their similarity.

无监督学习

Learning Rule — Unsupervised Learning

- The weights and biases are modified in response to network inputs only. *There are no target outputs available.*
- Most of these algorithms perform some kind of clustering operation. They learn to categorize the input patterns into a finite number of classes.

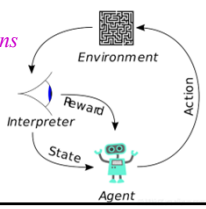


增强学习

Learning Rule — Reinforcement Learning

- Similar to supervised learning, except that, instead of being provided with the correct output for each network input
- *Only given a grade*
- The grade (score) is a measure of the network performance over some sequence of inputs
- Most suited to *control system applications*

E.g. Genetic Algorithm（GA）



感知器学习规则

Perceptron Learning Rule

- Update weights by:

$$\hat{w}_{ji} = w_{ji} + \eta(t_j - o_j)o_i$$

where η is the learning rate, t_j is the expected output for unit j

- Equivalent to rules:
  - If output is correct do nothing.
  - If output is high, lower weights on active inputs
  - If output is low, increase weights on active inputs

感知器学习

最简单的神经网络模型“单层感知器”就是由一个权值可调的TLU构成，可以采用有监督学习方式进行线性划分。

- 输入：给定正例集合P和反例集合N，对所有$x\in P, f(x)=1$，所有$x\in N, f(x)=0$
- 输出：w

感知器学习

- 权重初始化为

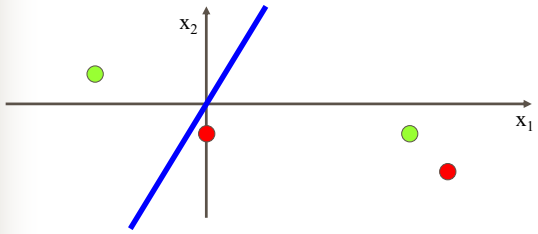
$$w = \sum_{x\in P} x - \sum_{x\in N} x$$

- 随机选择 $x \in P \cup N$
- 更新 $\hat{w} = w + \eta(t-o)x$（η为学习常数，t为期望输出，o为实际输出）
- 循环，直到所有训练示例的输出正确

Example

P = {(6,−1), (−3, 1)}

N= {(0,−1), (7,−2)} $\eta = 1$

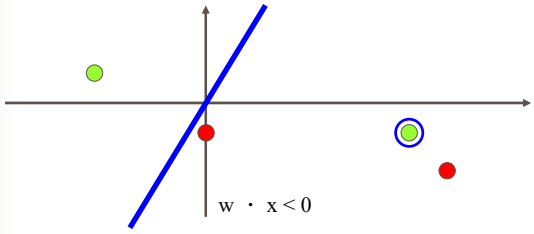


w = (6,−1) + (−3, 1) − (0,−1) − (7,−2) = (−4, 3)

Example

P = {(6,−1), (−3, 1)}

N= {(0,−1), (7,−2)} $\eta = 1$



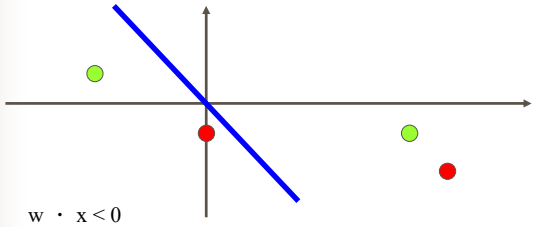
w · x < 0

w = w + (1-0)x =(-4, 3)+(6, -1)=(2, 2)

Example

P = {(6,−1), (−3, 1)}

N= {(0,−1), (7,−2)} $\eta=1$



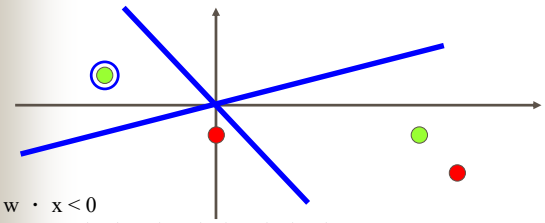
w · x < 0

w = w + (1-0)x =(-4, 3)+(6, -1)=(2, 2)

Example

P = {(6,−1), (−3, 1)}

N= {(0,−1), (7,−2)} $\eta=1$



w · x < 0

w = w + (1-0)x =(-4, 3)+(6, -1)=(2, 2)

w · x < 0

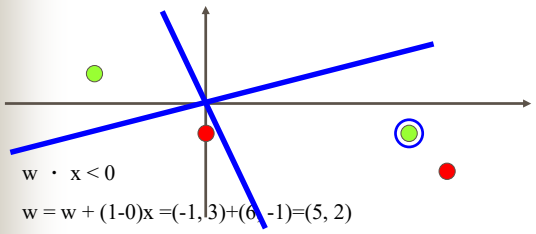
w = w + (1-0)x =(2, 2)+(-3, 1)=(-1, 3)

Example

Example

P = {(6,−1), (−3, 1)}

N= {(0,−1), (7,−2)} η=1



w · x < 0

w = w + (1-0)x =(-1, 3)+(6, -1)=(5, 2)

w · x < 0

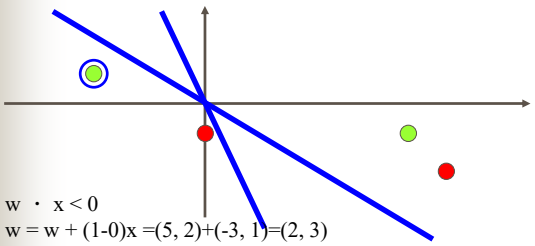
w = w + (1-0)x =(2, 2)+(-3, 1)=(-1, 3)

Example

Example

P = {(6,−1), (−3, 1)}

N= {(0,−1), (7,−2)} η=1



w · x < 0

w = w + (1-0)x =(5, 2)+(-3, 1)=(2, 3)

w · x < 0

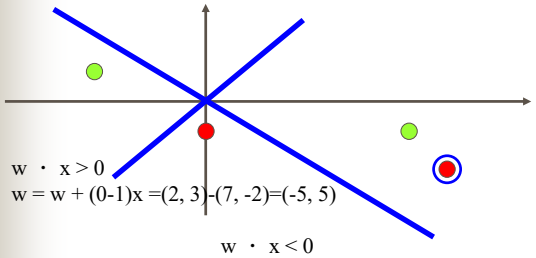
w = w + (1-0)x =(-1, 3)+(6, -1)=(5, 2)

Example

Example

P = {(6,−1), (−3, 1)}

N= {(0,−1), (7,−2)} η=1



w · x > 0

w = w + (0-1)x =(2, 3)-(7, -2)=(-5, 5)

w · x < 0

w = w + (1-0)x =(5, 2)+(-3, 1)=(2, 3)

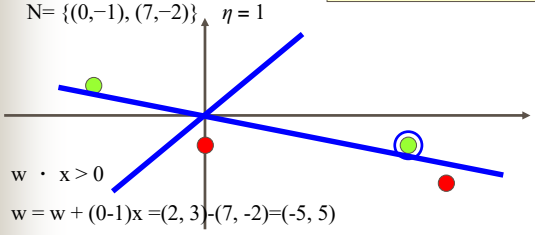
Example

Example

其他神经网络学习方法将在机器学习中具体介绍

P = {(6,−1), (−3, 1)}

N= {(0,−1), (7,−2)} η = 1



w · x > 0

w = w + (0-1)x =(2, 3)-(7, -2)=(-5, 5)

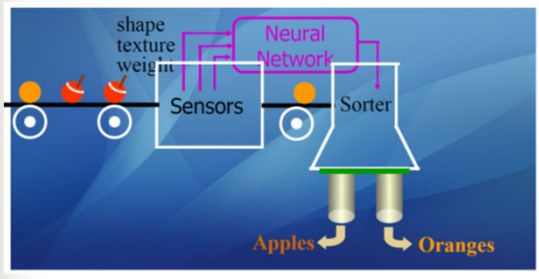
w · x < 0

w = w + (1-0)x =(-5, 5)+(6, -1)=(1, 4)

Example

Example

- Pattern Recognition
  - Problem Statement



Example

Example

Shape sensor: 1- - round, -1 - - elliptical

Texture sensor: 1- - smooth, -1 - - rough

Weight sensor: 1- - >1 pound, -1 - - <1 pound

$$\mathbf{p} = \begin{bmatrix} shape \\ texture \\ weight \end{bmatrix} \Rightarrow \mathbf{p}(apple) = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}, \quad \mathbf{p}(orange) = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix}$$

Example

Example

$$\mathbf{p} = \begin{bmatrix} shape \\ texture \\ weight \end{bmatrix} \Rightarrow \text{three-dimensional input } (R = 3)$$

$$n = \mathbf{W}\mathbf{p} + b,\ a = \text{hardlims}(n)$$

Choose the bias b and the elements of the weight matrix w so that the perceptron will can distinguish between apples and oranges.

Example

Example

$$\mathbf{p}(apple) = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} \quad \mathbf{p}(orange) = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} \Rightarrow \mathbf{W} = [0 \quad 1 \quad 0],\ b = 0$$

$$\textbf{Orange}: a = \text{hardlims}\left([0 \quad 1 \quad 0] \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} + 0 \right) = -1$$

$$\textbf{Apple}: a = \text{hardlims}\left([0 \quad 1 \quad 0] \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} + 0 \right) = 1$$

Example

Example

- What happens if we put a not-so-perfect orange into the classifier? That is to say, an orange with an elliptical shape is pass through the sensor.

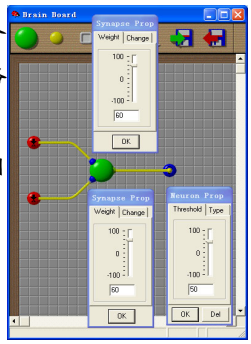
$$\mathbf{p} = \begin{bmatrix} shape \\ texture \\ weight \end{bmatrix} \Rightarrow \quad \mathbf{p}(orange) = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} \Rightarrow \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix}$$

$$a = \text{hardlims}\left([0 \quad 1 \quad 0] \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix} + 0 \right) = -1 \Rightarrow \textbf{orange}$$

Example

Example

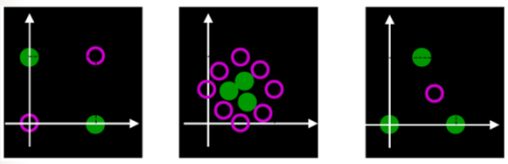
- 要求完成包括与、或、非等各种任务，完成任务的神经网络测试。
- 构造完成“或”逻辑的神经网络如图，所有权值阈值都可自行设置以完成任务



Limitations

Limitations

- The perception can be used to classify input vectors that can be separated by a *linear boundary*, like AND gate example.
  ⇨ *linearly separable* (AND, OR and NOT gates)
- Not linearly separable, e.g., XOR gate



神经网络应用

- 人工神经网络的特性及应用

A promising new generation of *information processing systems*, *usually operate in parallel*, that demonstrate the ability to learn（学习）, recall（记忆）, and generalize（概括，归纳，推广，泛化） from training patterns or data.