# AI-programmering

## Inlämningsuppgift 3

***Deadline:*** *4e Oktober - 23:59*

### 0. Introduktion

I denna inlämningsuppgift skall ni implementera K-nearest-neighbor (KNN) för att klassifiera MNIST datat, vilket är bilder på handskrivna siffror, och en perceptron för OR/XOR problemet. Ni kan använda valfritt programmeringsspråk, men implementationen skall vara eran från grunden, således får ni inte använda några biblitotek för maskininlärning. Mer information nedan.

Denna uppgift är uppdelad i två delar:

- För betyget 3 skall:
  - KNN implementeras för att klassifiera siffror i MNIST datat
  - Och en perceptron skall implementeras för att klassifiera OR-datat
- För betyget 5 skall eran perceptron implementering utökas till flera perceptroner för att lösa XOR-problemet.

### 0.1 Administrativt

- Någon form av metrik för prestanda av modell skall presenteras (accuracy, error etc.) som utskrift från programmet. Antingen i form av screenshots av terminal eller som textfil. Utkriftetn skall vara väl strukturerad och lätt att läsa och förstå.
- För inlämning skall källkod och ovanstående utksrift skickas in på Canvas
- Valfritt programmeringsspråk får användas
- Maskininlärningsbibliotek får ej användas (t.ex. Sci-kit, Torch, Tensorflow, Caffe, mlpack, OpenNN, OpenCV etc.)
  - Algoritmerna och ekvationer skall vara egenskrivna
  - Bibliotek för simpla matematiska operationer får användas (t.ex. standardbibliotek, Numpy etc.)
- MNIST datat kan hämtas från länken i canvas eller genom bibliotek
- Det finns inga krav för hög prestanda av modellen, men den ska vara bättre än att gissa på måfå

# 1. Uppgift 1 - För betyg 3

## 1.1 K nearest neighbor

**The pseudocode of classical KNN**

**Input:** X: training data, Y: class labels of X, K: number of nearest neighbors.
**Output:** Class of a test sample x.

***Start***
Classify (X,Y,x)
1. ***for*** each sample x ***do***
   Calculate the distance: $d(x, X) = \sqrt{\sum_{i=1}^{n}(x_i - X_i)^2}$
   ***end for***
2. Classify x in the majority class: $C(x_i) = argmax_k \sum_{X_j \epsilon KNN} C(X_j, Y_k)$

***End***

$X$ = hela datamängden

$x$ = en datapunkt från$X$

$Y$ = alla labels i datamängded (0-9 i detta fallet)

Ovan är pseudokoden för KNN som skall implementeras. KNN modellen som ni implementerar skall klassifiera vilken siffra som är skriven i bilderna från datat. Testningen skall ske på test-partitionen av datat. MNIST datat är relativt stort för denna algoritm, så ni kan behöva korta ner antalet bilder om det tar för lång tid att köra algoritmen. Datat är föruppdelat i träning och testning och är nedskalad i antal bilder och klasser från originaldatat, datat som finns på Canvas har bara siffrorna 1 och 6. Varje rad i datat är en bild på en siffra, där varje element i raden är en pixel av bilden. Det är 784 element (pixlar) i varje rad. Sista elementet i varje rad är label på datapunkten, alltså vilken siffra som bilden är. Ni kan läsa mer om datat här: https://www.tensorflow.org/datasets/catalog/mnist

## 1.2 Perceptron

**Algorithm 1:** The perceptron algorithm.

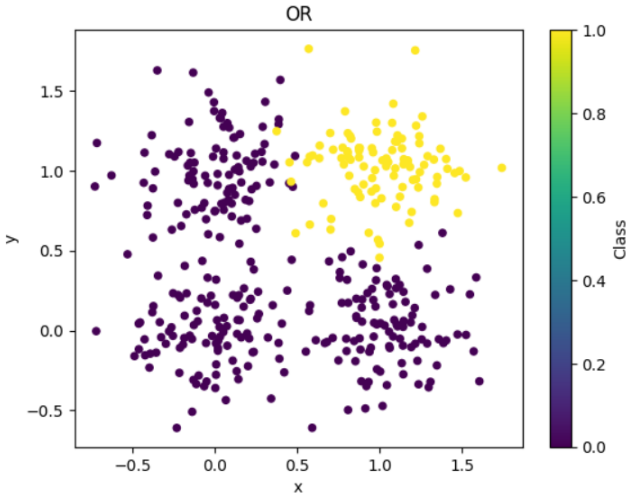
**Input:** $X \in \mathbb{R}^{n \times d}$, $\mathbf{y} \in \{-1, 1\}^n$, $\mathbf{w} = \mathbf{0}_d$, $b = 0$, max_pass $\in \mathbb{N}$
**Output:** $\mathbf{w}, b, mistake$

```
for t = 1, 2, ..., max_pass do
    mistake(t) ← 0
    for i = 1, 2, ..., n do
        if y_i(⟨x_i, w⟩ + b) ≤ 0 then
            w ← w + y_i x_i                    // x_i is the i-th row of X
            b ← b + y_i
            mistake(t) ← mistake(t) + 1
```

Ovan är pseudokod för en perceptron. Notera att ni måste inte utgå från denna pseudokoden, ni kan utgå från andra exempel sålänge det är en perceptron som implementeras. Modellen som ni implementerar skall klassifiera datapunkter i OR-datat (0 eller 1). Datat är uppdelad i train och test partitioner, och testningen skall ske på korrekt partition.

OR-datat innehåller 400 datapunker klustrade runt hörnen på en fyrkant likt OR problemet. Varje punkt har en x och y koordinat, samt label (0 eller 1). Datat ser ut som nedan:

| | x | y | Class |
|---|---|---|---|
| 1 | 0.0212937147120453… | 1.0456667215031104 | 0 |
| 2 | 0.9301209528176169 | 0.1981842235688581 | 0 |
| 3 | 0.08907331046138034 | -0.021407315798796… | 0 |
| 4 | -0.017005117085391… | 0.969465728012475 | 0 |
| 5 | -0.015028512793835… | 1.1955553946231752 | 0 |
| 6 | 0.9446833479673096 | 0.9787449208033525 | 1 |




Tips: Kom ihåg att en perceptron oftast tränas i flera epoker, där en epok är en träning genom hela träningsdatat. Börja med att köra i ca 10 epoker. För att göra det lätt för er kan ni räkna antalet felaktiga prediktioner under träningen och returnera denna lista efter en epok, man bör då kunna följa att antalet felaktiga prediktioner går ner efter varje epok (om man har gjort rätt). Tänk också på att det är vikterna som förbättras vid träningen, så det måste vara samma vikter som förbättras mellan varje epok. När perceptronen är färdigtränad så tar ni vikterna och sätter in i en perceptron där vikterna inte ändras. På så sätt kan ni prediktera på testdatat och jämföra modellens predikteringen med den faktiska klassen i datat.

2. **Uppgift 2** - För betyg 5

I denna uppgift skall ni expandera perceptron-modellen som ni skapade i uppgift 1. Ni skall implementera modellen som är beskriven i denna artikel [1]. Studera problemet noga. De viktigaste delarna från denna artikel presenteras nedan, men för bästa förståelse bör ni med fördel läsa artikeln (inkluderad i slutet), avsnitt 3 och 4.1 i artikeln är mest relevant.

2.1 **XOR-problemet**

Solution I

| $x_1$ | $x_2$ | $z_1$ | $z_2$ | $y$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 |

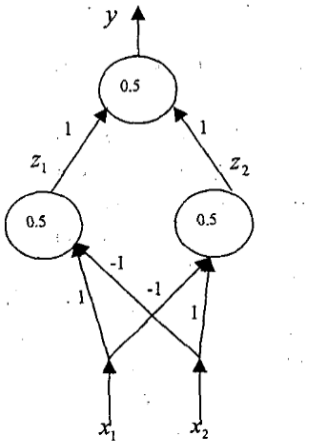



Fig. 5 XOR operation perceptron

Ovan presenteras en lösning på XOR-problemet från artikeln. Det är alltså ett nät av perceptroner (varje cirkel är en perceptron) där det gömda lagret ($Z_1$ och $Z_2$) har förbestämda labels som löser $y$. Eran uppgift är att ta implementationen ni gjorde i uppgift 1 och skapa ovan nät. Datat som ni skall träna nätet på är XOR-datat, som ser ut som OR-datat men där $Z_1$ och $Z_2$ är tillagda som kolumner. Nätet skall tränas på training partitionen och testas på testing partitionen. Detta nät bör också tränas i flera epoker like uppgift 1.

Tips: Se denna uppgift som 3 deluppgifter. Varje perceptron kommer ha sitt eget indata och ha en specifik label den skall skicka ut. Alltså, de två första parallella perceptronerna skall inte ha $y$ som label, utan $Z_1$ respektive $Z_2$ (vilket ses i diagrammet ovan). Den sista perceptronen

får då outputen från de tidigare två som sin input och $y$ som label. Den vänstra perceptronen kommer alltså att ha detta som sin datamängd

| $x_1$ | $x_2$ | $z_1$ |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 1 | 0 |

, där $Z_1$ är label och x,y är inputs. Likaså kommer den högra perceptronen ha liknande datamängd, men $Z_2$ som label. Den sista perceptronen kommer ha

| $z_1$ | $z_2$ | $y$ |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 0 | 0 | 0 |

som datamängd med $Z_1$ och $Z_2$ som input och $y$ som label. Se det som att de första två parallella perceptronerna skapar indata för den sista perceptronens $y$.

[1]. **Analysis and Study of Perceptron to Solve XOR Problem**

Proceedings of
the 2nd International Workshop on
Autonomous Decentralized System

# Analysis and Study of Perceptron to Solve XOR Problem

**Zhao Yanling, Deng Bimin, Wang Zhanrong**
*Southwest Jiaotong University, Chengdu 610031,Sichuan, China*
*E-mail: ylzhao@home.swjtu.edu.cn*

## Abstract

*This paper tries to explain the network structures and methods of single-layer perceptron and multi-layer perceptron. It also analyses the linear division and un-division problems in logical operation performed by single-layer perceptron. XOR is linear un-division operation, which cannot be treated by single-layer perceptron. With the analysis, several solutions are proposed in the paper to solve the problems of XOR. Single-layer perceptron can be improved by multi-layer perceptron, functional perceptron or quadratic function. These solutions are designed and analyzed.*

## 1. Introduction

Artificial Neural Network (ANN)[1], constructed by the human beings based on a knowledge of Neural Networks of their brains, is a functional Neural Network, which is an information handling system taking after the structure and functions of human brain. Compared with John Von Neumann's series computer, it has higher parallelism, stronger non-linear mapping ability, better error tolerance and association memory function and greater self-learning ability. ANN has been widely used in mode recognition, information handling, associative memory, solution optimizing and self-adapting control.

ANN can be structurally divided into two categories: feedforward Neural Network and feedback Neural Network. The former is mainly adopted in the field of system discriminating and mode recognition, etc; the latter emphasizes on associative memory and optimization, etc. And the former can be further classified as multi-layer perceptron, single-layer perceptron and functional link. Among them perceptron plays an important role in the problem of linearity divisibility and non-linearity divisibility.

As we all know, according to Boolean logic, especially in duality calculation, there are operators as "AND", "OR", "NOT", "NOT.AND", "NO.TOR", "XOR" and "X.NOT.OR", and only "XOR" and "X.NOT.OR" are linearity non-divisible. It has been a hot issue to materialize and construct the two operators in Neural Network. The limitations of perceptron are discussed in reference [2]. The functional combining net structure is introduced in references [3] [4]. In this paper, the authors first give an introduction of networking and methods of multi-layer perceptron and single-layer perceptron, and analyse the problem of linearity divisibility in the way of logical operation by perceptron, and then propose, with some inter-comparisons, several solutions on how to utilize the perception to solve XOR problem.

## 2. Perceptron

### 2.1 Single-layer perceptron

As for the network structure of single-layer perceptron, see the following figure 1.

Among these: $x_i (i = 1,2,\cdots,n)$ represents the input of the neuron $i$ at the input level; $y$ represents the output of the neuron at the output level; $\omega_i$ represents the



0 -7803 -7624 -2/02/$17.00 ©2002 IEEE

connection progression value of the neuron $i$ at the input

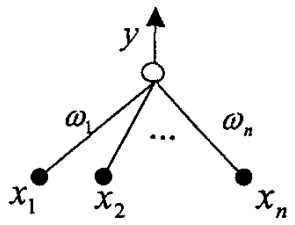



Fig. 1 Single-layer perceptron

level and the neuron at the output level.

The output of the neuron fills the formula (1), (2).

$$y = f(\sum_{i=1}^{n} \omega_i x_i - \theta) \qquad (1)$$

$$f(\mu) = \begin{cases} 1 & \mu \geq 0 \\ 0 & \mu < 0 \end{cases} \qquad (2)$$

Among these, $f(\cdot)$ is the activation function. Here we pick unit function or the sign function Sgn $(\cdot)$ or S function Sigmoid $(\cdot)$; $\theta$ is the threshold value.

The learning algorithm of the single-layer perceptron:

(1) Put the preliminary value: make $t = 0$, give $\omega_i(t)$ $(i = 1,2,\cdots,n)$ and $\theta(t)$ a smaller ram value ($\neq 0$) respectively.

(2) Input a learning sample $X = (x_1, x_2, \cdots, x_n)^T$ and its expected output $d$.

(3) Account the actual output and error:

$$y = f\left[\sum_{i=1}^{n} \omega_i(t) x_i - \theta(t)\right]$$

$$e = (d - y)^2$$

(4) Amend all the progression value and the threshold value.

$$\omega_i(t+1) = \omega_i(t) + \eta(d - y)y(1 - y)x_i$$

$$(i = 1,2,\cdots,n)$$

$$\theta(t+1) = \theta(t) + \eta(d - y)y(1 - y)$$

Among these $\eta$ is the learning step width which picks the number between 0~1.

(5) Shift to (2) until progression value and the threshold value keep the same to all the learning samples, that is the convergence value is got.

### 2.2 Multi-layer perceptron

See the following network diagram of multi-layer perceptron:

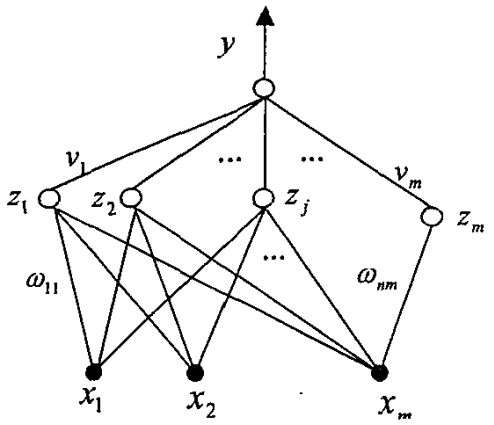



Fig. 2 Multi-layer perceptron

Among these, $x_i$ $(i = 1,2,\cdots,n)$ indicates input of the No. $i$ neuron at the input level. $z_j (j = 1,2,\cdots,n)$, output of No. $j$ neuron at the concealed level. $y$, neuron output at the output level. $\omega_{ij}$, connection progression value between the No. $j$ neuron at the concealed level and the No. $i$ neuron at the output neuron. $v_j$ represents the connection progression value of the neuron $j$ at the concealed level and the neuron at the output level.

Neuron output at various levels satisfies following formulas: (3), (4), (5)

$$z_j = f\left[\sum_{i=1}^{n} \omega_{ij} x_i - \theta_j\right] \qquad (3)$$

$$y = f\left[\sum_{j=1}^{m} v_j z_j + \theta\right] \qquad (4)$$

$$f(\mu) = \begin{cases} 1 & \mu \geq 0 \\ 0 & \mu < 0 \end{cases} \qquad (5)$$

## 3.Perceptron classification and XOR problem

In order to be general, we'll think about the perceptron of single neuron in the section 2.1.

The adjustment of the connection progression value can make perceptron's reaction to a group of vectors achieve the objective output of 0 or 1. This can be explained by making diagrams in input vector space. Picking two input variables, $x_1$, $x_2$, we can get figure 3:

The condition $\omega_1 x_1 + \omega_2 x_2 - \theta \geq 0$ will divide the input plane into two parts. When the connection progression value and the threshold value change, the dividing line will move or turn round, but still keeps in a straight line. The threshold value divides the space of vectors into several areas, which enable the perceptron to be capable of classifying the input vectors and input samples. Perceptron cannot realize the corresponding relation between the input and the output arbitrarily, or it can't solve the arbitrary logical operation. It can only solve the problem of the linear division mentioned in figure 4, that is to say, preceptron cannot classify the problem of linear un-division.

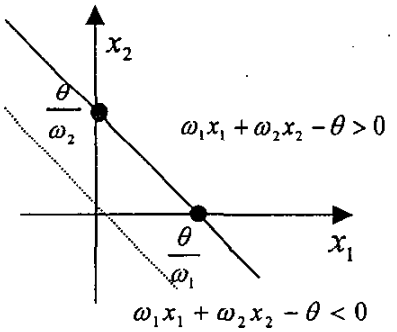



Fig. 3 Input vector plane map

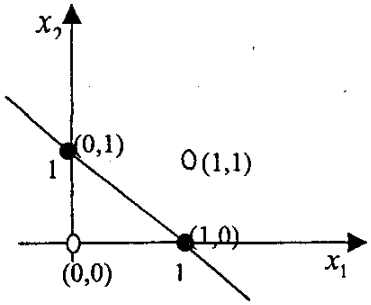



Fig. 4 XOR problem sketch map

XOR is a typical problem in linear un-division. What XOR means in logical operation is: when the two inputs are the same 1 or the same 0 in binary system, the output is 0; when the two inputs are 1 or 0 respectively, the output is 1. XOR requires dividing the four points in a plane with a straight line as indicated in figure 4. Obviously it is impossible. In general condition, it is impossible to solve the problem of XOR with regular single-layer perceptron.

## 4. Analysis of perceptron solution to solve XOR problem

According to analysis above, the performance and learning ability of single-layer perceptron is limited. In general condition, the simple single-layer perceptron can't realize XOR, but only the multi-layer perceptron can solve the problem of XOR. In order to solve the problem of XOR, we propose several solutions: multi-layer perceptron, functional link perceptron, and single-layer perceptron can be improved by quadratic function.

### 4.1 Multi-layer perceptron to solve XOR problem

It is impossible to solve the problem of XOR within the limitation of ordinary single-layer perceptron. The simplest and most effective way is to add hiding layers to change the single-layer net structure into multi-layer net structure by using multi-layer perceptron. Here we will list some solutions. We take the output of each layer as in the formula in section 2.1, namely:

$$y = f\left(\sum_{i=1}^{n} \omega_i z_i - \theta\right)$$

$$z_i = f\left(\sum_{j=1}^{2} v_j x_j - \theta_j\right)$$

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

$n$ is the number of hiding layer of neuron.

Solution 1

| $x_1$ | $x_2$ | $z_1$ | $z_2$ | $y$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 |

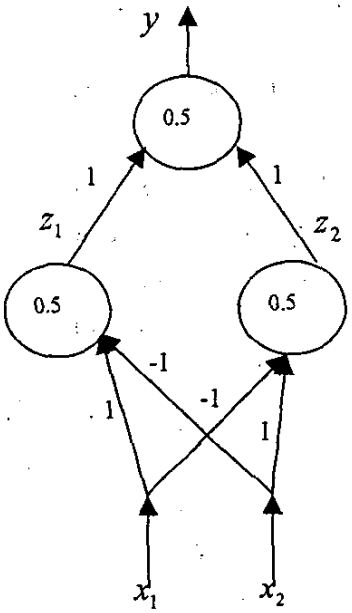



Fig. 5 XOR operation perceptron

Solution 2

| $x_1$ | $x_2$ | $z_1$ | $z_2$ | $y$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 |

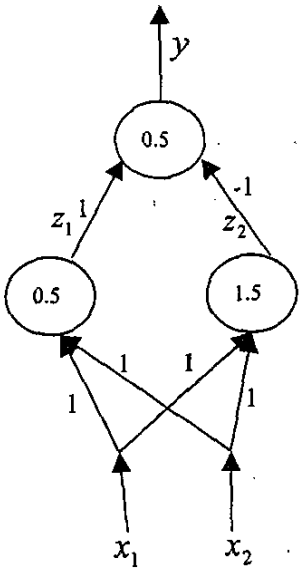



Fig. 6 XOR perceptron of new $\omega, \upsilon, \theta_i$

Solution 3

| $x_1$ | $x_2$ | $z$ | $y$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |

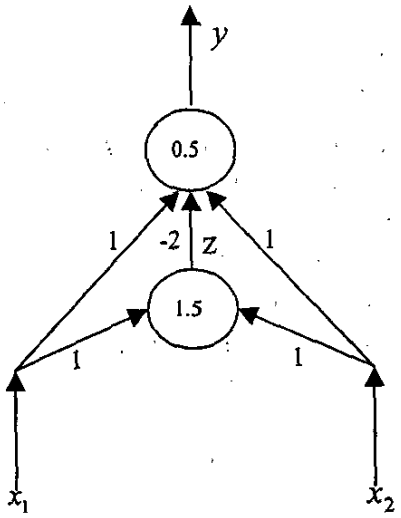



Fig. 7 Perceptron of one hiding neuron

Solution 4

| $x_1$ | $x_2$ | $z_1$ | $z_2$ | $z_3$ | $y$ |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 0 |

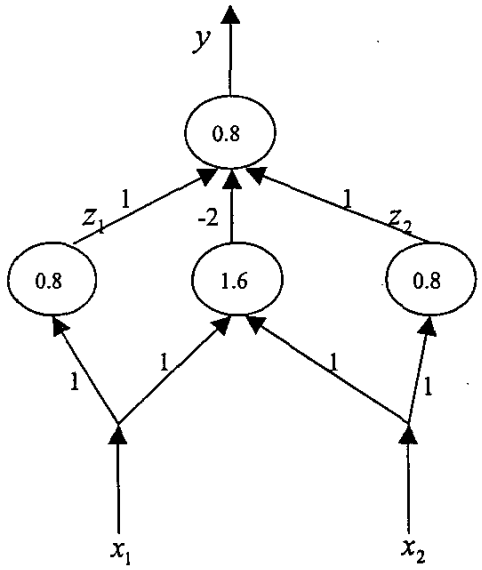



Fig. 8 Perceptron of three hiding neurons

In solution 4, there are three neurons in the middle layer, in theory, there is no limit on the number of neurons ($n \geq 1$). However, it is unnecessary as the number will increase working load and has a direct impact on the speed of constringency. which results in making the task more difficult to fulfill. In fact, solution 1,2,3 is the simplified forms of solution 4.

In addition, the activation function $f$, the threshold value $\theta$、$\theta_j$ and the weight $\omega_{ij}$ are adjustable in the previous four solutions. That is to say, there are a lot of combination projects that will be enumerated there.

## 4.2 Functional combining perceptron

In some references [3] [4], the item of high rank is introduced, called Functional Combination. Through functional combining perceptron, many problems can be solved by single-layer net. Figure 9 shows:

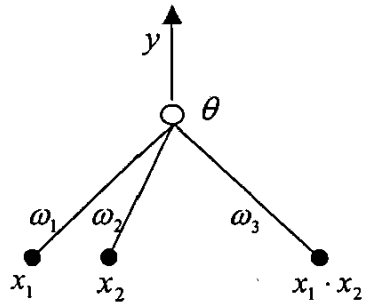



Fig. 9 Functional combnating network

On the basis of simple single-layer perceptron, the introduction of $x_1 \cdot x_2$ means that one input is added, and this input is not new variable in nature but the production of the mutual influence or encouragement of $x_1$ and $x_2$. So the output of this net is

$$y = f(x_1\omega_1 + x_2\omega_2 + x_1x_2\omega_3 - \theta)$$

It can solve the problem of XOR. In fact, make $\omega_1 = \omega_2 = 1, \omega_3 = -2, \theta = 1$, and we have:

$$y = f(x_1 + x_2 - 2x_1x_2 - 1)$$

Rotating 45° , the coordinate is changed to be a new one: $x_1', x_2'$, the quadratic curve $x_1 + x_2 - 2x_1x_2 = 1$ is changed to be:

$$-\frac{(x_1' - \sqrt{2}/2)^2}{(\sqrt{2}/2)^2} + \frac{(x_2')^2}{(\sqrt{2}/2)^2} = 1$$

It's a hyperbola. It inputs two groups of models into {(0,0), (1,1)} {(1,0), (0,1)} and divides into two types: "o" type and "1" type.

## 4.3 Quadratic function perceptron

The structures of quadratic function perception are similar to the single-layer perceptron, while the differences lie in which the activation function is quadratic function. Output of neural network should be satisfied with the

following formula:

$$y = f\left[\sum_{i=1}^{n} \omega_i x_i - \theta\right]$$

$$f(\mu) = \mu^2$$

In quadratic function perceptron, the output expected value is zero and big enough value L. To the problem of duality XOR, we value L=1. Let's assume the learning algorithm as the follows:

(1) Make it originate: Let $t = 0$ , give $\omega_i(t)(i = 1,2,\cdots,n)$ and $\theta(t)$ each a smaller random value which isn't zero.

(2) In put a learning sample $X = (x_1, x_2, \cdots, x_n)^T$ and its expected output "$d$".

(3) Calculate the actual output

$$y = f[\sum_{i=1}^{n} \omega_i(t) x_i - \theta(t)]$$

(4) Modify each right value and threshold value

When $d = 0$:

$$\omega_i(t+1) = \omega_i(t) - \eta\sqrt{y}x_i \qquad (i = 1,2,\cdots,n)$$

$$\theta(t+1) = \theta(t) - \eta\sqrt{y}$$

When $d = L$:

$$\omega_i(t+1) = \omega_i(t) + \eta\sqrt{y}x_i \quad (i = 1,2,\cdots,n)$$

$$\theta(t+1) = \theta(t) + \eta\sqrt{y}$$

Here $\eta$ is learning step width.

Turn (5) to (2), till the neural network is stable and unchanged for all learning samples.

From the structures and learning algorithm, we know: when $d = 0$, each learning process of the quadratic function perceptron is a process of getting crest value point of the quadratic function. The best step width we choose is:

$$\eta = \frac{1}{1 + \sum_{i=1}^{n} x_i^2}$$

XOR is linear un-division operation that cannot be treated by the novel single-layer perceptron. Quadratic function perceptron is capable of learning XOR problems. The essence of this improved quadratic function perceptron lies in the following fact: neuron activation function employing quadratic function to replace unit function (or Sigmoid function), select expectation and the most optimized learning step width. Simulation is also feasible by utilizing other functions forms.

## 5.Conclusion

XOR is linear un-division operation, which cannot be treated by single-layer perceptron. Six solutions are proposed to solve the problem of XOR. Four in which are realized by means of multi-layer perceptron. General expressions have been provided, threshold value and activation function are also adjustable, many potential solutions are possibly involved. Multi-layer neural network can always solve the problems of XOR or "X.NOT.OR", and can implement any elements Boolean function and logical calculations, but at the cost of complication of the system. The functional perceptron and quadratic function perceptron used to solve the problems of XOR belong to improved sing-layer perceptron which are characterized by more powerful learning functions and faster convergence speed, compared with traditional single-layer perceptron.

## References


[1] Jinfan, *Neural Computational Intelligence Basic*, Southwest Jiaotong University Press, 2000.

[2] Congshuang, "The Analysis, Limitations and Enlargement of perceptron", *Automation Panorama*, No.3, 2000, pp. 34-36.

[3] Hanmei, "Discussion of XOR Problem", *Journal of Hebei University*, Vol.19, No.1, Mar. 1999, pp. 24-27.

[4] Li Hongxing, "Mathematical Neural Networks(III)", *Journal of Beijing Normal University(Natural Science)*, Vol.33, No.3, Sep. 1997, pp. 305-311.