THE CHINESE UNIVERSITY
OF HONG KONG (SHENZHEN)
School of Science and Engineering

Tutorial - 11

Fundamental of Artificial Intelligence **CSC3180**

1. Answer the following questions:

(a) Why Artificial Neural Network (ANN)? (L12: P4-8)

Challenge: Modern computer

- Traditional
- Powerful
- Sequential
- Logic-based digital

➢ Less successful for other types of problems like common human activity.

(b) What's ANN? What's the assumption of ANN (L12: P10-14)? And list some applications of ANN. (L12: P16)

Definition:

- An information-processing system that has certain performance characteristics in common biological neural networks
- Developed as generalizations of mathematical models of human neural biology, based on the assumptions:
    - Information processing occurs at neurons
    - Signals are passed between neurons
    - Each connection link has an associated weight
    - An activation function is applied to each neurons

Assumption:

- The processing element receives many signals which may be modified by a weight at the receiving synapse
- The processing element sums the weighted inputs
- The neuron transmits a single output to many other neurons (fanout)
- Information processing is local
- A synapse's strength may be modified
- Neurotransmitters for synapses may be excitatory or inhibitory
- The processing element receives many signals which may be modified by a weight at the receiving synapse
- The processing element sums the weighted inputs
- The neuron transmits a single output to many other neurons (fanout)
- Information processing is local
- A synapse's strength may be modified
- Neurotransmitters for synapses may be excitatory or inhibitory
- ANN can be characterized by
    - its pattern of connections (architectures)
    - its method of determining the weights (called training or learning)
    - its activation function
- ANN consists of a large numbers of simple processing elements called neurons, units, cells or nodes
- ANN is motivated by a desire to try both to understand the brain and to emulate some of its strengths

Application:

- Signal Processing
  - Suppress noise on a telephone line
- Control
  - Learn how to steer the truck for the trailer to reach a dock
- Pattern Recognition
  - Automatic recognition of handwritten characters
  - Handle large variation in sizes, positions & styles of writing
- Medicine
  - Store a large no. of medical records, each includes information on symptoms diagnosis, & treatment for a particular case
- Speech Production and Recognition
  - Read new one after training English words
  - Train speaker-independent recognition
- Business
  - Mortgage assessment work (Use past experience to train ANN to provide more consistent & reliable evaluation of mortgage applications & determine whether the applicant should be given a loan)

(c) Please understand some ANN designs (L12: P18)

- Setting the Weights
  - Supervised training
  - Unsupervised training
  - Fixed weights
- Common Activation Functions
  - Identity function
  - Binary step function
  - Binary sigmoid (0,1)
  - Bipolar sigmoid (-1,1)

(d) What's the Hebb net? How to optimize the Hebb net? What's the application? (L12: P30-33)

Hebb net: a single-layer (feed-forward) ANN trained by the Hebb rule is defined as Hebb net. For Hebb net, data is represented in bipolar form. The weight is defined as: $w_i(new) = w_i(old) + x_i y$.

Application: Fit AND function, OR function and classify two-dimensional input patterns.

(e) What is perceptron learning? Which difference between Hebb net and perceptron? (L12: P35-37)

- More powerful learning rule than Hebb rule
- Its iterative learning procedure can be proved to converge to the correct weights
- The original perceptron had three layers of neurons – sensory units, associator units, and a response unit
- Use binary step function with an arbitrary, but fixed, threshold as its activation function

$$f(y\_in) = \begin{cases} 1 & if\ y_{in} > \theta \\ 0 & if\ -\theta \le y\_in \le \theta \\ -1 & if\ y_{in} < -\theta \end{cases}$$

- If an error occurred for a particular training input pattern, the weight would be changed according to the formula

$$w_i(new) = w_i(old) + \alpha(t - f(y\_in))\, x_i$$

where t is the target and α is the gain or step size.

(f) What's BP? Please describe the BP algorithm. (L12: P43-48)

Back-Propagation:

- Play a major role in the reemergence of ANNs as a tool to solve a wide range of problems;
- BP also known as generalized delta rule.
- A gradient descent method to minimize the total squared error of the output computed by the net

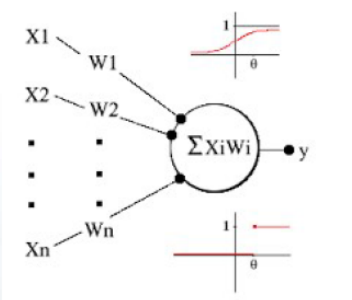
Step 0. Initialize weights. (Set to small random values).
Step l. While stopping condition is false, do Steps 2-9.
Step 2. For each training pair, do Steps 3-8.

Feedforward:

Step 3. Each input unit $(X_i, i = 1, \dots, n)$ receives input signal x_i and broadcasts this signal to all units in the layer above (the hidden units).

Step 4. Each hidden unit $(Z_j,\ j = 1, \dots, p)$ sums its weighted input signals, $z\_in_j = b_j + \sum x_i v_{ij}$, applies its activation function to compute its output signal, $z_j = f(z\_in_j)$, and sends this signal to all units in the layer above (output units). Here, taking Sigmoid function as example,

$$z_j = f(z\_in_j) = \frac{1}{1+e^{-z\_in_j}}$$



Step 5. Each output unit $(Y_k, k = 1, \dots, m)$ sums its weighted input signals, $y\_in_k = b_k + \sum z_j w_{jk}$, and applies its activation function to compute its output signal, $y_k = f(y\_in_k)$.

Backpropagation of error:

Step 6. Each output unit $(Y_k, k = 1, \dots, m)$ receives a target pattern corresponding to the input training pattern, computes its error information term, $\delta_k = (t_k - y_k) f'(y\_in_k)$. *Here, takes Sigmoid activation function*, for example, $f(y\_in_k) = \frac{1}{1+e^{-y\_in_k}}$. Its derivate is $f'(y\_in_k) = f(y_{in_k})\left(1 - f(y_{in_k})\right)$. Calculate its weight correction term, $\Delta w_{jk} = \alpha \delta_k z_j$, calculate its bias correction, $\Delta b_k = \alpha \delta_k$, and send δ_k to units in the layer below. Here, α is the learning rate, which is the step size for updating the parameters.

Tips: The derivate of $f(x) = \frac{1}{1+e^{-x}}$.

$$f'(x) = \frac{-(1+e^{-x})'}{(1+e^{-x})^2} = \frac{e^{-x}}{(1+e^{-x})^2} = \frac{1}{1+e^{-x}} \cdot \frac{e^{-x}}{1+e^{-x}} = f(x)\frac{1+e^{-x}-1}{1+e^{-x}}$$

$$= f(x)\left(\frac{1+e^{-x}}{1+e^{-x}} - \frac{1}{1+e^{-x}}\right) = f(x)(1 - f(x))$$

Step 7. Each hidden unit (Z_j, $j = 1, \dots, p$) sums its delta inputs (from units in the layer above), $\delta\_in_j = \sum \delta_k w_{jk}$, multiplies by the derivative of its activation function to calculate it error information term, $\delta_j = \delta\_in_j f'(z\_in_j) = \delta_{in\,j} f(z\_in_j)(1 - f(z\_in_j))$, calculates its weight correction term, $\Delta v_{ij} = \alpha \delta_j x_i$, and calculates its bias correction term, $\Delta b_j = \alpha \delta_j$.

Update weights and biases:

Step 8. Each output unit ($Y_k, k = 1, \dots, m$) updates its bias and weights ($j = 1, \dots, p$) :

$$w_{jk}(new) = w_{jk}(old) + \Delta w_{jk}$$
$$b_k(new) = b_k(old) + \Delta b_k$$

Each hidden unit (Z_j, $j = 1, \dots, p$) updates its bias and weights ($i = 1, \dots, n$)

$$v_{ij}(new) = v_{ij}(old) + \Delta v_{ij}$$
$$b_j(new) = b_j(old) + \Delta b_j$$

Step 9. Test stopping condition.

2. **Set that we have a dataset with 4 samples in two classes just as in the Table 1 and a Hebb net for it just as in Figure 1. Initialize the weight w_1, w_2 and b with 0. Please update the weight with Hebb rule and calculate the weight after training.**

x_1	x_2	y
0	1	1
1	0	1
-1	0	-1
0	-1	-1

Table 1: Samples for Hebb net.

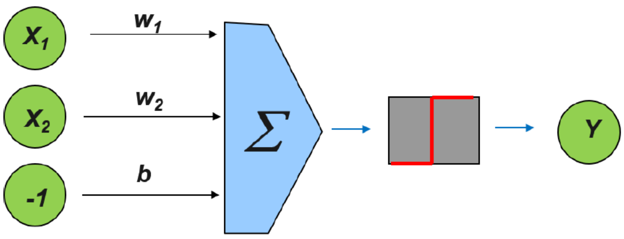


Figure 1: Hebb net

(1) Input data point (0,1,1): $w_1=0+0*1=0$; $w_2=0+1*1=1$; $b=0+1=1$.

(2) Input data point (1,0,1): $w_1=0+1*1=1$; $w_2=1+0*1=1$; $b=1+1=2$.

(3) Input data point (-1,0,-1): $w_1=1+-1*-1=2$; $w_2=1+0*-1=1$; $b=2-1=1$.

(4) Input data point (0,-1,-1): $w_1=2+0*-1=2$; $w_2=1+-1*-1=2$; $b=1-1=0$.

After training, $w_1=2$, $w_2=2$, $b=0$. The threshold function will set the output large than 0 to be 1 and the output smaller than 0 to be -1. Finally, the Hebb net can classify the given dataset.

3. **Table 2 is a dataset with 4 samples in two classes. And a very simple network with only one single neuron is designed to fit y in Figure 2. Initialize the weight w_1, w_2 and b with 0. The loss function is defined as $L = \frac{1}{4}\sum_{i=1}^{4} \|y - f(x_1(i), x_2(i))\|^2$. $x_1(i)$ and $x_2(i)$ represent x_1 and x_2 in row i of Table 2. Set the learning rate to be 0.1. Please update the weight for 1 iteration with the BP algorithm.**

x_1	x_2	y
0	1	1
1	0	1
-1	0	0
0	-1	0

Table 2: Samples for BP.

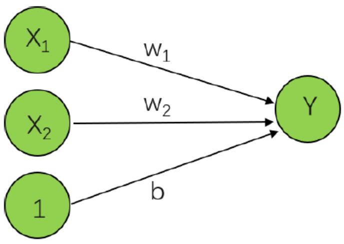


Figure 2: Network with one single neuron.

Forward: Calculate $f(x_1,x_2)$ and the loss:

(x_1,x_2,y)	$f(x_1,x_2)=w_1x_1+w_2x_2+b$	$\|\| y-f(x_1,x_2)\|\|^2$
(0,1,1)	0*0+0*1=0	1
(1,0,1)	0*1+0*0=0	1
(-1,0,0)	0*-1+0*0=0	0
(0,-1,0)	0*0+0*-1=0	0

L=1/4*(1+1+0+0)=0.5

Backward: Calculate the derivate:

$$\begin{aligned}\frac{\partial L}{\partial f} &= \frac{1}{2}(f(x_1,x_2)-y)\\ \frac{\partial f}{\partial w_1} &= x_1\\ \frac{\partial f}{\partial w_2} &= x_2\\ \frac{\partial f}{\partial b} &= 1\end{aligned}$$

Gradient of w_1,w_2,b:

$$\begin{aligned}\frac{\partial L}{\partial w_1} &= \frac{\partial L}{\partial f}\frac{\partial f}{\partial w_1} = \frac{1}{2}(f(x_1,x_2)-y)x_1\\ \frac{\partial L}{\partial w_2} &= \frac{\partial L}{\partial f}\frac{\partial f}{\partial w_2} = \frac{1}{2}(f(x_1,x_2)-y)x_2\\ \frac{\partial L}{\partial b} &= \frac{\partial L}{\partial f}\frac{\partial f}{\partial b} = \frac{1}{2}(f(x_1,x_2)-y)\end{aligned}$$

(x_1,x_2,y)	*Gradient of w_1*	*Gradient of w_2*	*Gradient of b*
(0,1,1)	0	-0.5	-0.5
(1,0,1)	-0.5	0	-0.5
(-1,0,0)	0	0	0
(0,-1,0)	0	0	0

Gradient of w1 = 0-0.5+0+0=-0.5;

Gradient of w2 = -0.5+0+0+0=-0.5;

Gradient of b = -0.5-0.5+0+0=-1;

Update the weight:

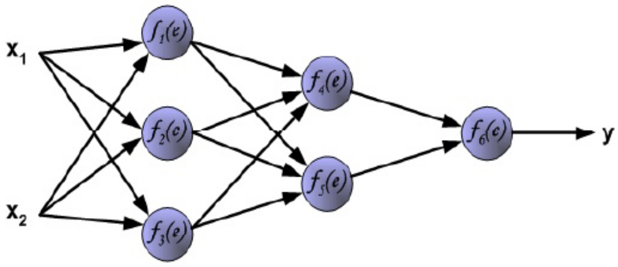
w1= 0-0.1*-0.5=0.05;

w2=0-0.1*-0.5=0.05;

b=0-0.1*-1=0.1.

4. A neural network with two inputs, x1 and x2, and one output, y, is given in Lecture 12, which has three layers for XOR applications.

1) Please understand the three stages: Feedforward, Backpropagation of Error, and Update Weights & Biases.



2) There are ten characters, A, B, …, J (each 7x9 binary pixels). Please design a neural network which has both 63 input and 63 output layers to implement data compression. Note that the tolerance could be 0.2.