0.1 Neural Network

- *Algorithm:* Neural Network(algo. 1)
- *Input:* A tensor
- *Complexity:* For a n-layer neural network,of which each layer has $n_i (i \in 1, 2, \cdots, n)$, the complexity is $\mathcal{O}(\sum_{i=1}^{n-1} n_i n_{i+1})$.
- *Data structure compatibility:* Matrix
- *Common applications:* Artificial intelligence

Problem. Neural Network

Neural network is a kind of mathematical model imitating the mechanism of biological neural network.

Description

Linear classification machine

In the early development of artificial intelligence, people want to use a simple predicting machine to implement intelligence. The simple predicting machine is actually a linear classification machine.

$$y = Ax + B \tag{0.1.1}$$

where x is input and A, B are linear parameter. However, this machine cannot solve non-linear problem like **XOR**[1] problem(Fig. 1). After studying biological neural network, scientist find the combination of activation

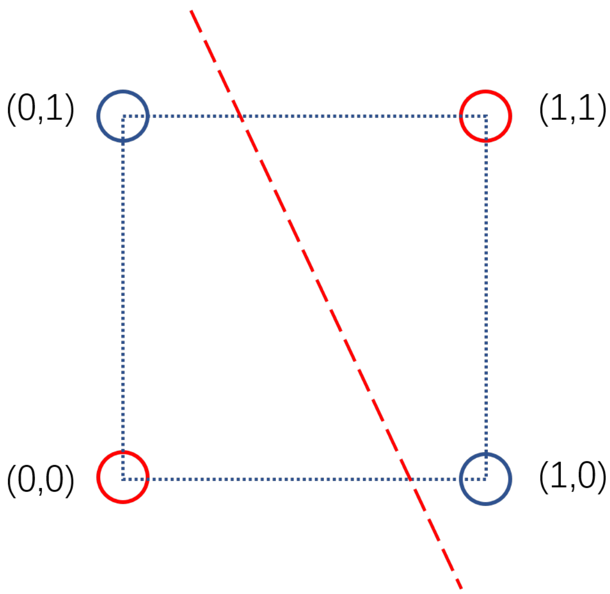

Figure 1: XOR problem

function and connection model can solve non-linear problem. That is modern neural network.

Forward propagation

In convenience, we will use a 3-layer(Fig. 2) to demonstrate the theory. The first layer is an input layer, which has the same dimension as the input tensor. The second layer is a hidden layer, whose dimension is decided by

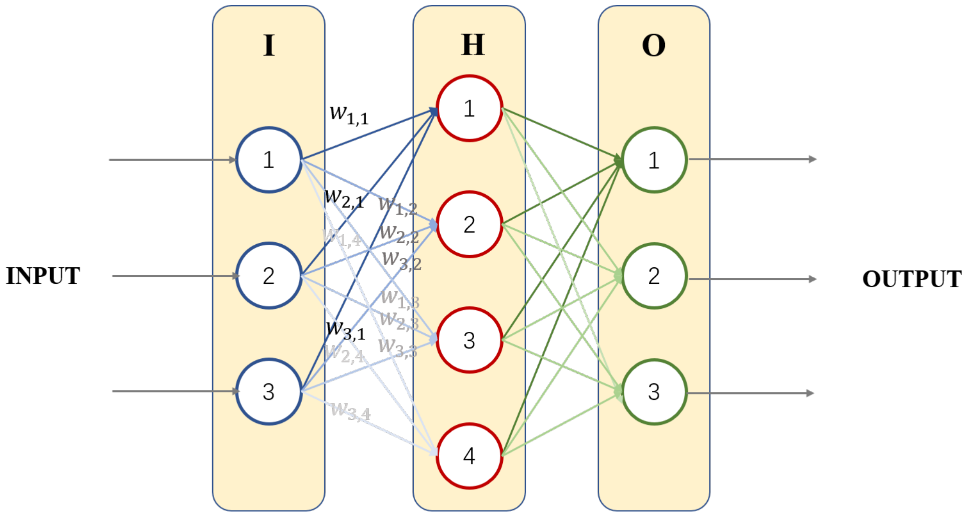

Figure 2: Structure of a 3-layer neural network

user. The last one is an output layer, which has the same dimension as the number of categories. We use I, H, O to denote values of input, hidden and output layer. The feed forward process can be calculated as following equation. For example, H_1 can be calculated as

$$h_1' = I_1 \cdot w_{1,1} + I_2 \cdot w_{2,1} + I_3 \cdot w_{3,1} \tag{0.1.2}$$

$$h_1 = A(h_1) \tag{0.1.3}$$

where $A(x)$ means activation function. Some common activation functions are

1. sigmoid

$$y = \frac{1}{1 + x^{-x}} \tag{0.1.4}$$

2. tanh

$$y = tanh(x) \tag{0.1.5}$$

3. Relu

$$y = \begin{cases} x & x \geq 0 \\ 0 & x < 0 \end{cases} \tag{0.1.6}$$

We can also use matrix to simplify our expression. The input matrix is

$$I = \begin{pmatrix} i_1 \\ i_2 \\ i_3 \end{pmatrix} \tag{0.1.7}$$

The hidden and output matrix are similar. The weight matrix is

$$W_{I,H} = \begin{pmatrix} w_{1,1} & w_{1,2} & w_{1,3} \\ w_{2,1} & w_{2,2} & w_{2,3} \\ w_{3,1} & w_{3,2} & w_{3,3} \end{pmatrix} \tag{0.1.8}$$

So the forward propagation of input and hidden layer can be represent as

$$H = A(W_{I,H}^{\mathsf{T}} I) \tag{0.1.9}$$

In the same way the hidden and output layer is

$$O = A(W_{H,O}^{\mathsf{T}} H) \tag{0.1.10}$$

Back propagation

We first define loss function L to represent the error between ground truth and the output. The most commonly used is square error.

$$L = \sum_n (t_n - o_n)^2 \tag{0.1.11}$$

To update weight, we must calculate

$$\frac{\partial L}{\partial w_{j,k}} = \frac{\partial}{\partial w_{j,k}} (T_k - O_k)^2 \tag{0.1.12}$$

We then use chain rule. The activation function is sigmoid here.

$$\begin{aligned}
\frac{\partial L}{\partial w_{j,k}} &= \frac{\partial L}{\partial O_k} \cdot \frac{\partial O_k}{\partial w_{j,k}} \\
&= -2(t_k - o_k) \cdot \frac{\partial O_k}{\partial w_{j,k}} \\
&= -2(t_k - o_k) \cdot \frac{\partial}{\partial w_{j,k}} \sigma(\sum_j w_{j,k} \cdot o_j) \\
&= -2(t_k - o_k) \cdot \sigma(\sum_j w_{j,k} \cdot o_j)(1 - \sigma(\sum_j w_{j,k} \cdot o_j)) \cdot o_j
\end{aligned}$$

The update function is

$$w_{j,k}^{(new)} = w_{j,k}^{(old)} - \alpha \cdot \frac{\partial L}{\partial w_{j,k}} \tag{0.1.13}$$

α is learning rate which should be tuned.

Algorithm 1: Neural network

Input : Training set $\{(I_i, T_i)\}, i = 1, 2 \cdots N$, Rounds R
Output: Output vector O of sample I

```
for r = 1 to R do
    for i = 1 to N do
        Forward propagation
        Calculate loss function L
        Back propagation
    end for
end for
return O
```

References.

[1] Marvin Minsky and Seymour Papert. *Perceptrons: An Introduction to Computational Geometry.* Cambridge, MA, USA: MIT Press, 1969 (cit. on p. 1).