# Multilayer Neural Networks

# Topics of this lecture

- What is a neural network?
- Model of one neuron.
- Learning rules for one neuron.
- Layered neural network.
- Learning of multilayer neuron network.

- ニューラルネットとは？
- ニューロンモデル
- ニューロンの学習則
- 階層型ニューラルネット
- 階層型ニューラルネットの学習

# What is a brain?

- Brain is the CPU for a human or animal that controls the whole body.
- Brain is a huge and complex network of approximately $10^{11}$ neurons, with each neuron connects approximately $10^4$ to other neurons.
- Although the switching speed of each neuron is slow, the whole brain can make complex decisions quickly, and can even “think”.

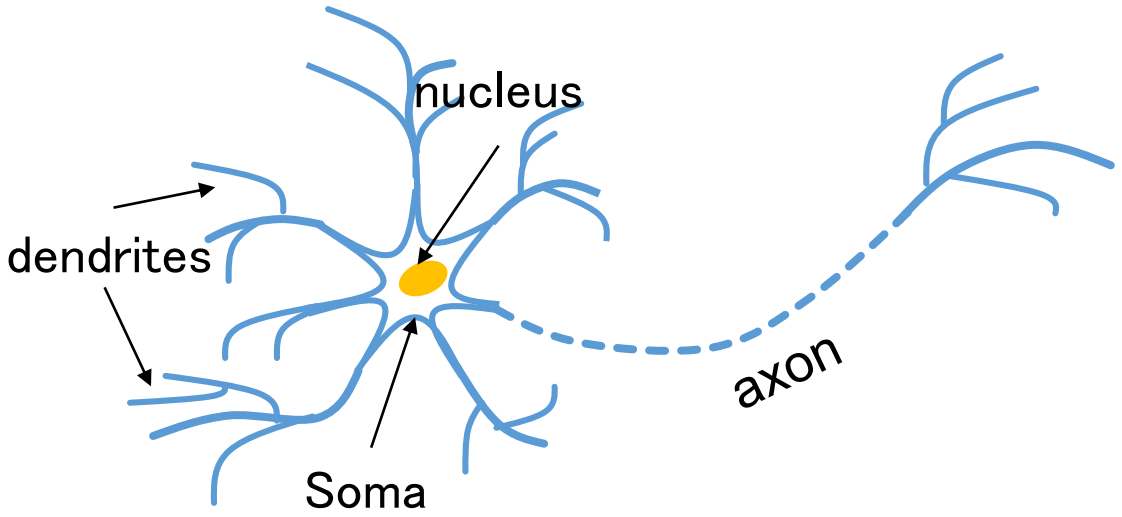
# Brief history of neural network study

- 1943: McCullochとPitts: First mathematic model of neuron
- 1958: Rosenblatt: Perceptron - Single layer NN
- 1982: Hopfield: Associative memory - Recurrent NN
- 1986: Rumelhart: Back Propagation algorithm - First boom of NN
- 1995: Y. LeCun, Y. Bengio, et al.: Convolutional neural network
- 1996: R4-rule: structural learning of NN-based network
- 2006: G. E. Hinton, et al.: Deep belief nets.
- 2006: Neural network tree (NNTree): Tree-based structural learning
- 2016: Boom of deep learning-based AI.


- *Q. F. Zhao and T. Higuchi, "Evolutionary learning of nearest neighbor MLP," IEEE Trans. on Neural Networks, Vol. 7, No. 3, pp. 762-767, 1996.*
- *Q. F. Zhao, "Inducing NNC-Trees with the R4-rule," IEEE Trans. on Systems, Man, and Cybernetics - Part B: Cybernetics, Vol. 36, No. 3, pp. 520-533, 2006.*

# Mechanism of a neuron

- A neuron consists of a soma, many dendrites and an axon.
- Pulses from other neurons are inputted to the soma from the dendrites via synapses, and are integrated there.
- The neuron sends a pulse to other neurons through the axon when the potential of the soma is higher than a threshold.



# McCulloch and Pitts Model

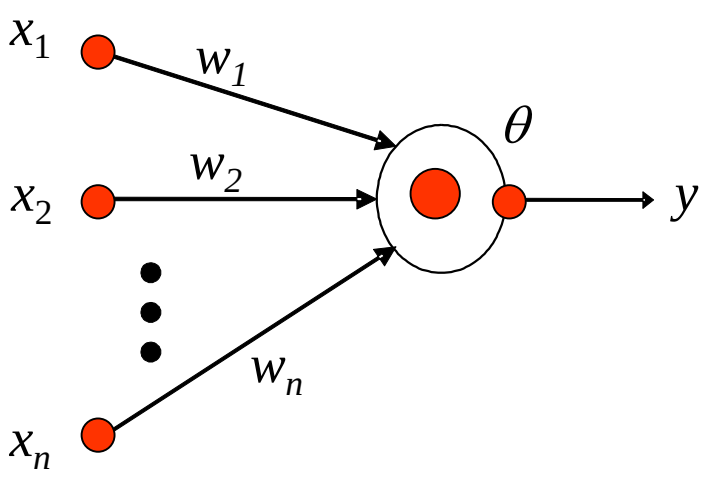
Suppose that $x_1, x_2, \cdots, x_n$ are the inputs, $w_1, w_2, \cdots, w_n$ are the connection weights, and $y$ is the output of the neuron. Then,

$$y = g(u), \quad u = \sum_{i=1}^{n} w_i x_i - \theta$$

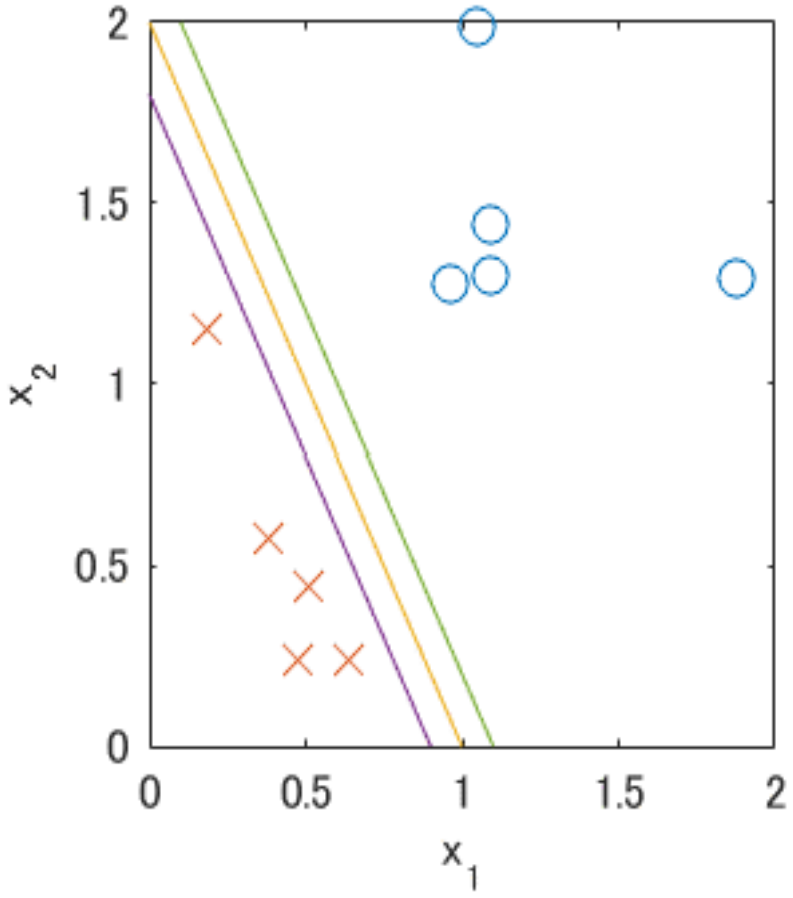
**$u$ is called the effective input of the neuron**

where $\theta$ is the threshold or bias, $g(u)$ is the activation function of the neuron, and is given by

$$g(u) = \begin{cases} 1 & u \geq 0 \\ 0 & u < 0 \end{cases}$$



# Example 5.4 pp. 99-100




**There are two classes of patterns. The patterns can be separated by one line defined by**

$$2x_1 + x_2 - 2 = 0$$

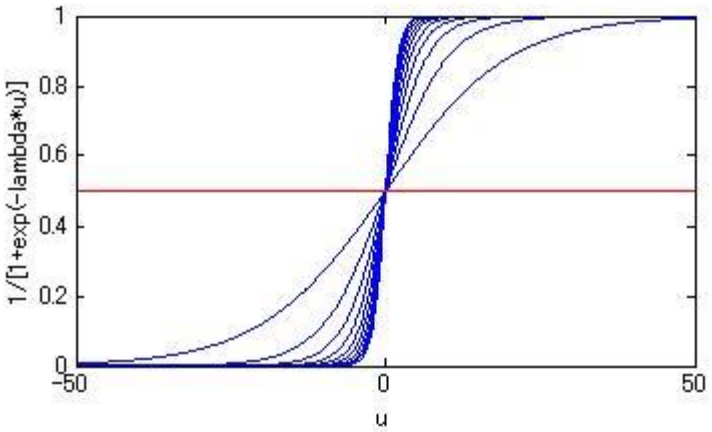
**Thus, if we define w1=2, w2=1, and θ=-2, the neuron can be used to classify all patterns correctly.**

# Continuous activity functions

(1) Unipolar sigmoid function

$$g(u) = \frac{1}{1+\exp(-\lambda u)}$$

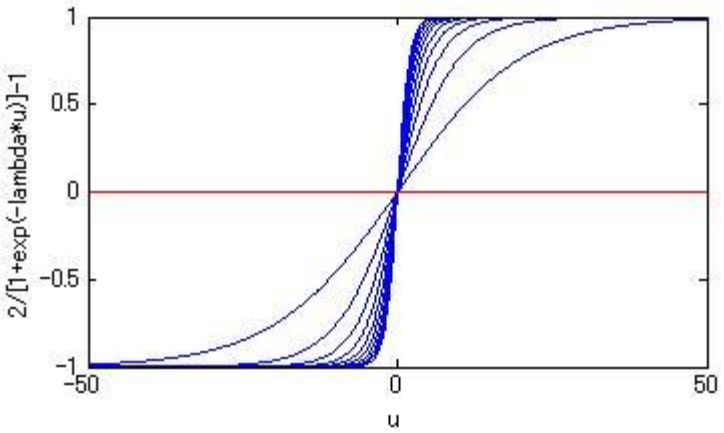
$$\Rightarrow \quad g'(u) = g(u)[1-g(u)]$$




(2) Bipolar sigmoid function

$$g(u) = \frac{1-\exp(-\lambda u)}{1+\exp(-\lambda u)}$$

$$\Rightarrow \quad g'(u) = \frac{1}{2}[1-g^2(u)]$$

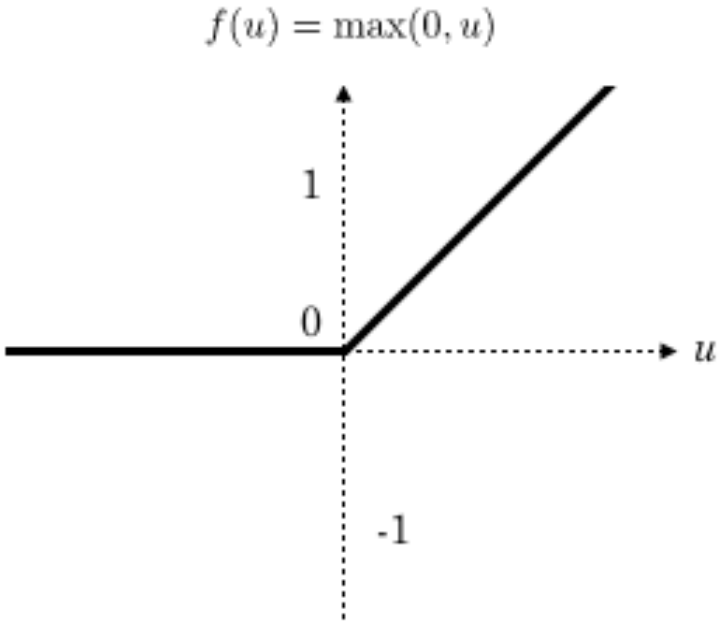


# Continuous activity functions

- Rectified linear unit (ReLU)

$$f(u) = \max(0, u)$$

- Where $u$ is the effective input.
- ReLU is often used for deep learning, to avoid the gradient vanishing problem.



# General rule for neuron learning

Based on a training set, we can train a neuron using the following rule :

$$\mathbf{w}^{new} = \mathbf{w}^{old} + cr\mathbf{x}$$

where **x** is a training datum selected from the training set (usually one - by - one), $c$ is a learning constant, and $r$ is the learning signal.

# Perceptron learning rule

If we define the learning signal $r$ by $r = d - y$, we get the perceptron learning rule. That is,

$$\mathbf{w}^{new} = \mathbf{w}^{old} + c(d - y)\mathbf{x}$$

where $d$ and $y$ are, respectively, the desired output (or teacher signal) and the actual output. It is known that the above rule converges if a solution exists.

# Program for perceptron learning

```
Initialization();
while(Error>desired_error){
   for(Error=0,p=0; p<n_sample; p++){
      y=FindOutput(p);
      Error+=0.5*pow(d[p]-y, 2.0);
      r=d[p]-y;
      for(i =0;i<I;i++){
         w[i]=w[i] + r*c*x[p][i];
      }
   }
}
```

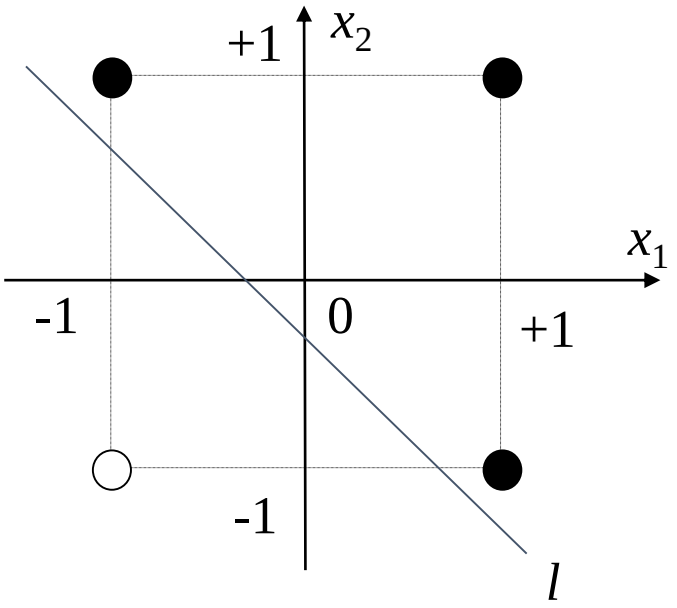
# Remarks

- During learning, each datum is usually selected one by one from the training set $\Omega$.
- Training a neuron on each datum of $\Omega$ once is an epoch or a learning cycle.
- In each epoch, the error is calculated as follows:

$$E = \sum_{i=1}^{|\Omega|} | d^i - y^i |$$

# Example 7.1 p. 142

- Four patterns are given by (1,1),(1,-1),(-1,1) and (-1,-1).
- The corresponding teacher signals are 1, 1, 1, and -1.
- The problem is to find a neuron using the perceptron learning rule, to separate the four patterns into two classes.




| Epochs | Connection weights | Number of errors |
|---|---|---|
| 1 | 0.117988 0.106789 0.748825 | 1 |
| 2 | 1.117988 1.106789 -0.251175 | 0 |

# The delta learning rule

If the activation function is differentiable, we can use the following delta - learning rule :

$$\mathbf{w}^{new} = \mathbf{w}^{old} + c(d - y)g'(u)x$$

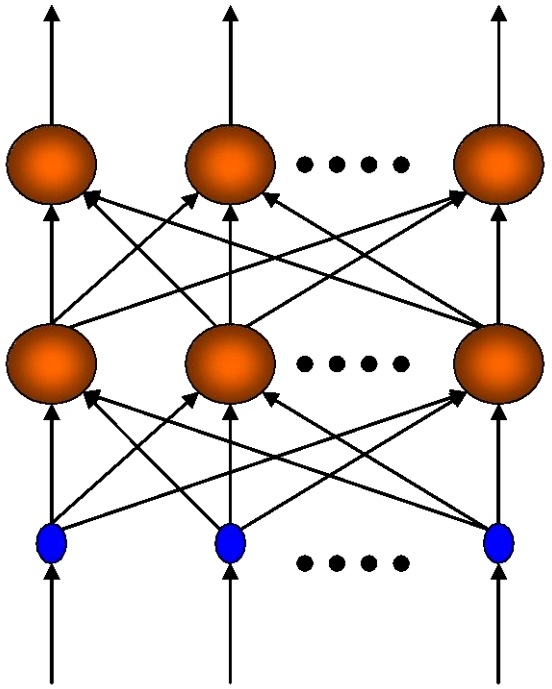
where $u = w^t x =< w, x >= \sum_{i=1}^{n} w_i x_i$ is the effective input, and $g'(u)$ is the first order direvative of the activiation function at $u$. Clearly, the learning signal $r$ is defined by $(d - y)g'(u)$. It is known that the above rule convergies to the *mean squared error* solution.

# Program for delta learning rule

```
Initialization();
  while(Error>desired_error){
       for(Error=0,p=0; p<n_sample; p++){
       FindOutput(p);
       Error+=0.5*pow(d[p]-y,2.0);
       for(i=0;i<I;i++){
      delta=(d[p]-y)*(1-y*y)/2;
      w[i]=w[i]+c*delta*x[p][i];
        }
     }
}
```

# Multilayer perceptron

- The right figure is a multilayer neural network or multilayer perceptron (MLP).
- There are three layers: input layer, hidden layer, and output layer.
- There can be more than two hidden layers.
- MLP is known as a general approximator that be used to approximate any given function.



# Notations

- $x_i$ : The *i-th* input ($i$=1,2,…,$I$)
- $y_j$ : The output of the *j-th* hidden neuron ($j$=1,2,…,$J$)
- $z_k$ : The output of the *k-th* output neuron ($k$=1,2,…,$K$)
- $v_{ji}$ : The weight from the *i-th* input to the *j-th* hidden neuron
- $w_{kj}$ : The weight from the *j-th* hidden neuron to the *k-th* output neuron

# Learning of output layer neurons

- The weights of each output neuron can be determined directly using the delta learning rule.
- For example, the k-th output neuron
  - Select a training datum x;
  - Find the outputs $y_j$ of all hidden neurons;
  - Find the output $z_k$ of the k-th output neuron;
  - Update the weights of the k-th output neuron based on $z_k$, $d_k$, and $y_j$ (j=1,2,…,J)

# Learning of hidden neurons

Error back propagation (BP) is a method for finding the error signal of the hidden neurons from that of the output neurons. The error signal of the k-th output neuron is given by

$$\delta_{zk} = (d_k - z_k) f'(u_k), \qquad \text{where } u_k = \sum_{j=1}^{J} w_{kj} y_j$$

The error signal of the $j$-th hidden neuron is then calculated by

$$\delta_{y_j} = f'(u_j) \sum_{k=1}^{K} \delta_{zk} w_{kj}, \qquad \text{where } u_j = \sum_{i=1}^{I} v_{ji} x_i$$

Using the error signals, we can update the weights of the j-th hidden neuron as follows:

$$v_{ji}^{new} = v_{ji}^{old} + c\delta_{y_j} x_i$$

# Summary of BP-based learning

- Step 1: Initialize the weights
- Step 2: Reset the total error
- Step 3: Get a training example *x* from the training set, calculate the outputs of the hidden neurons and those of the output neurons, and update the total error
- Step 4: Calculate the error signals as follows:

$$\delta_{zk} = (\boldsymbol{d}_k - \boldsymbol{z}_k)(1 - \boldsymbol{z}_k^2)/2$$

$$\delta_{y_j} = \left( \sum_{k=1}^{K} \delta_{zk} \boldsymbol{w}_{kj} \right) (1 - \mathrm{y}_\mathrm{j}^2)/2$$

# Summary of BP-based learning

- Step 5: Update the weights as follows:

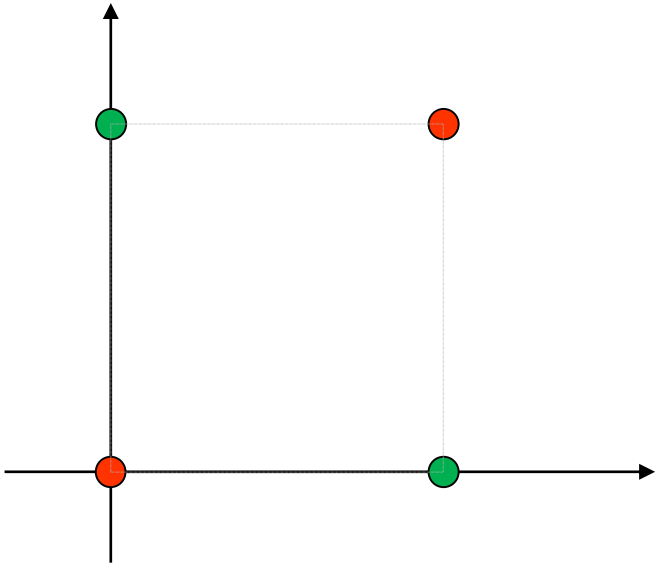
$$\boldsymbol{w}_{kj}^{new} = \boldsymbol{w}_{kj}^{old} + \eta\delta_{zk}\boldsymbol{y}_j \quad \text{for } \boldsymbol{k} = 1,2,\cdots,\boldsymbol{K};\ \boldsymbol{j} = 1,2,\cdots,\boldsymbol{J}$$

$$\boldsymbol{v}_{ji}^{new} = \boldsymbol{v}_{ji}^{old} + \eta\delta_{y_j}\boldsymbol{x}_i \quad \text{for } \boldsymbol{j} = 1,2,\cdots,\boldsymbol{J};\ \boldsymbol{i} = 1,2,\cdots,\boldsymbol{I}$$

- Step 6: See if all training examples have been used. If NOT, return to Step 3
- Step 7: See if the total error is smaller than the desired value. If NOT, reset the total error, and return to Step 2; otherwise, terminate.

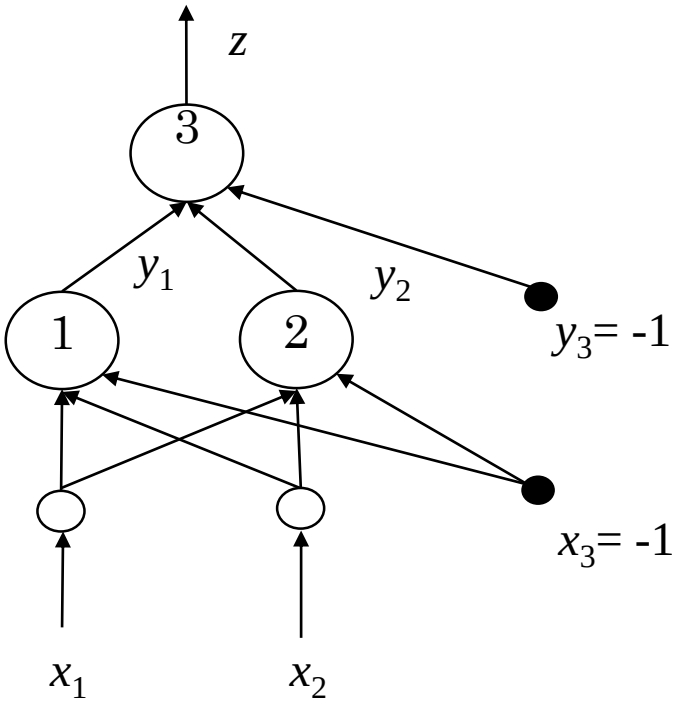
# Example 7.3 p. 147

- Consider the XOR problem that classifies the vertices of a square into two classes.
- That is, (0,0) and (1,1) belong to the negative class; and (1,0) and (0,1) belong to the positive class.
- This is a simple problem, but cannot be solved by using a single layer neural network.



# The network structure

- The right figure shows the structure of the MLP.
- There are two input neurons, two hidden neurons, and one output neuron.
- The input neurons are just registers.



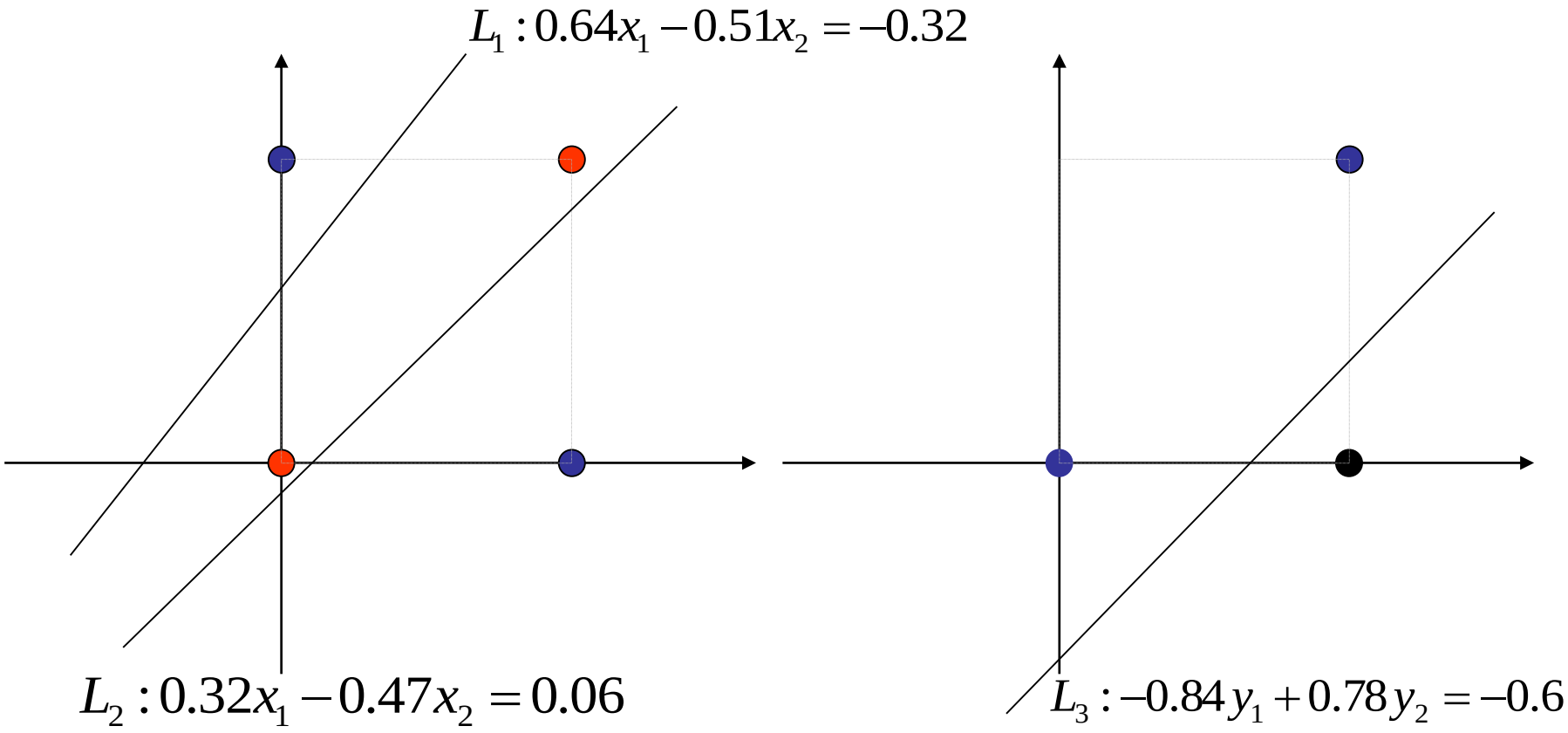
# Results of BP

```
  ................
Error[1073]=0.001008
Error[1074]=0.001003
Error[1075]=0.000998

The connection weights in the output layer:
-0.839925 0.782646 -0.602153

The connection weights in the hidden layer:
0.638360 -0.509153 -0.316935
0.319389 -0.472554 0.068378
```

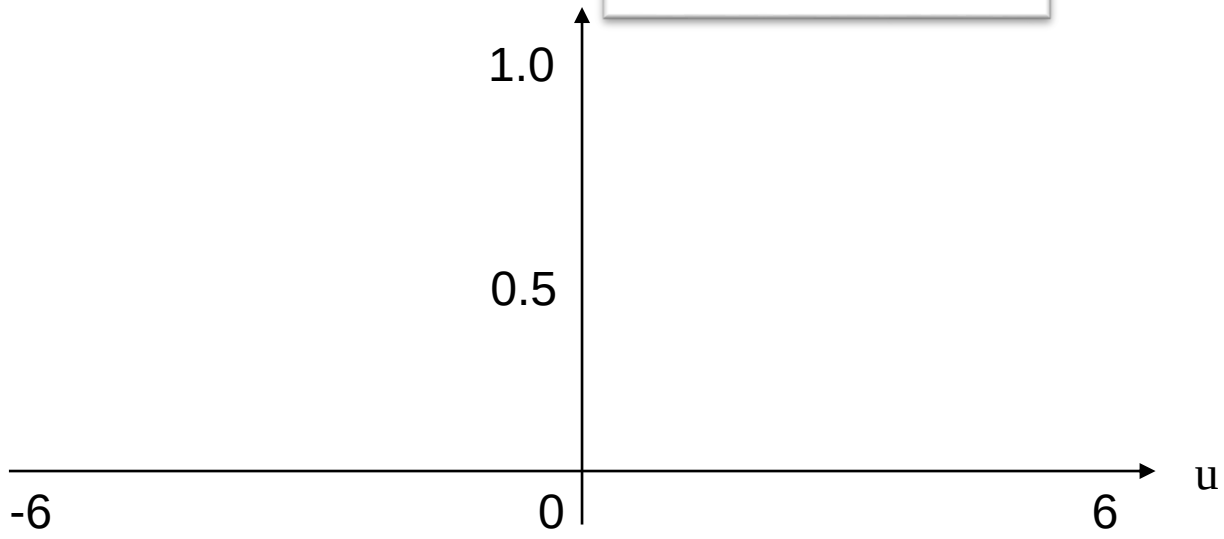
# Physical meaning of the result



# Homework for lecture 12 (1)

- Draw the unipolar sigmoid function, print out this page, and submit it to the TA during the exercise class. $\lambda$ is supposed to be 1.

$$f(u) = \frac{1}{1+\exp(-\lambda u)}$$



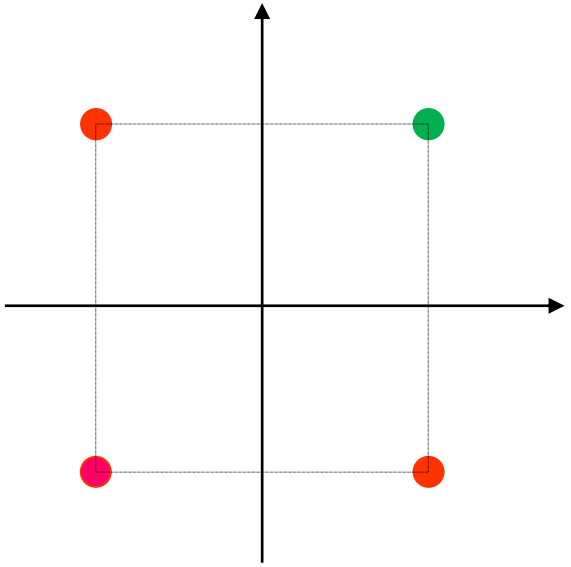
# Homework for lecture 12 (2)

- Given in the right is the truth table of the 4-bit parity check problem.
- Write a program that can train an MLP based on the BP algorithm.
- Suppose that the number of layers is 3, and unipolar sigmoid function (λ = 1) is used as the activation function.

| Input vector | | | | Teacher |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 |

# Quizzes for lecture 12 (1)

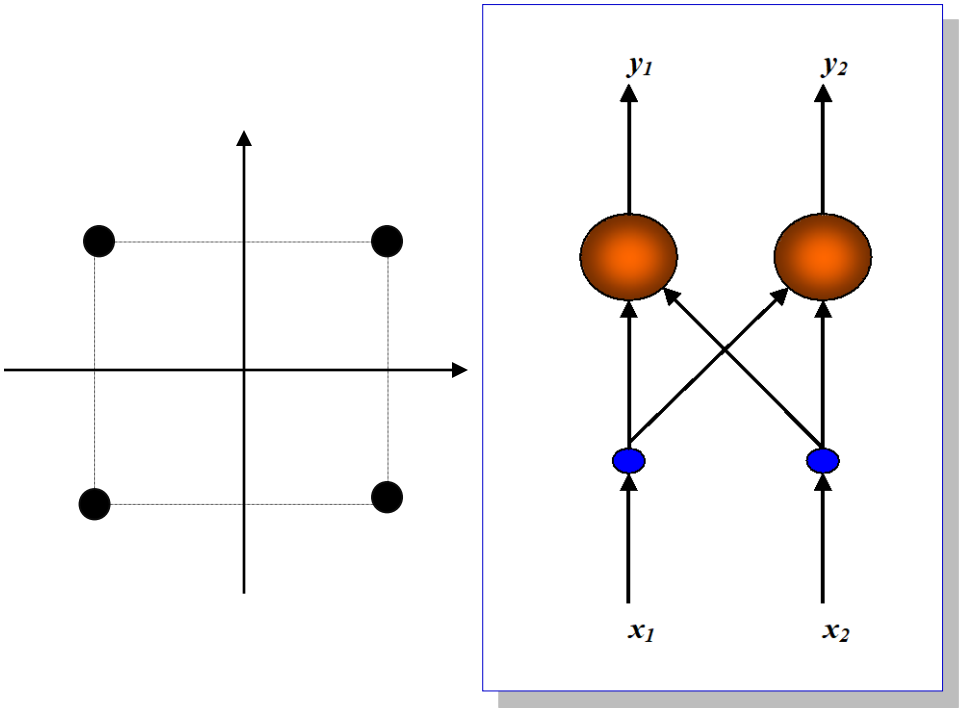
- Find the weights $w_1$, $w_2$, and $w_3$ of a neuron for classifying the four vertices of a square into two groups.
- The data and the teacher signals are given below:
  - (-1,-1), (1,-1), (-1,1), (1,1)
  - -1, -1, -1, 1
- There are infinitely many solutions. Please just provide one solution.



$$w_1 x_1 + w_2 x_2 - w_3 = 0$$

# Quizzes for lecture 12 (2)

- Suppose that the data and their teacher signals are given by:
  - (-1,-1), (1,-1), (-1,1), (1,1)
  - (-1,1), (1,-1), (1,1), (1,1)
- Find two neurons to solve the problem. Just provide one solution.




$$Neuron1: w_{11}x_1 + w_{12}x_2 - w_{13} = 0$$

$$Neuron2: w_{21}x_1 + w_{22}x_2 - w_{23} = 0$$