Neural Network and Deep Learning

Multi-Layer Perceptron (MLP)

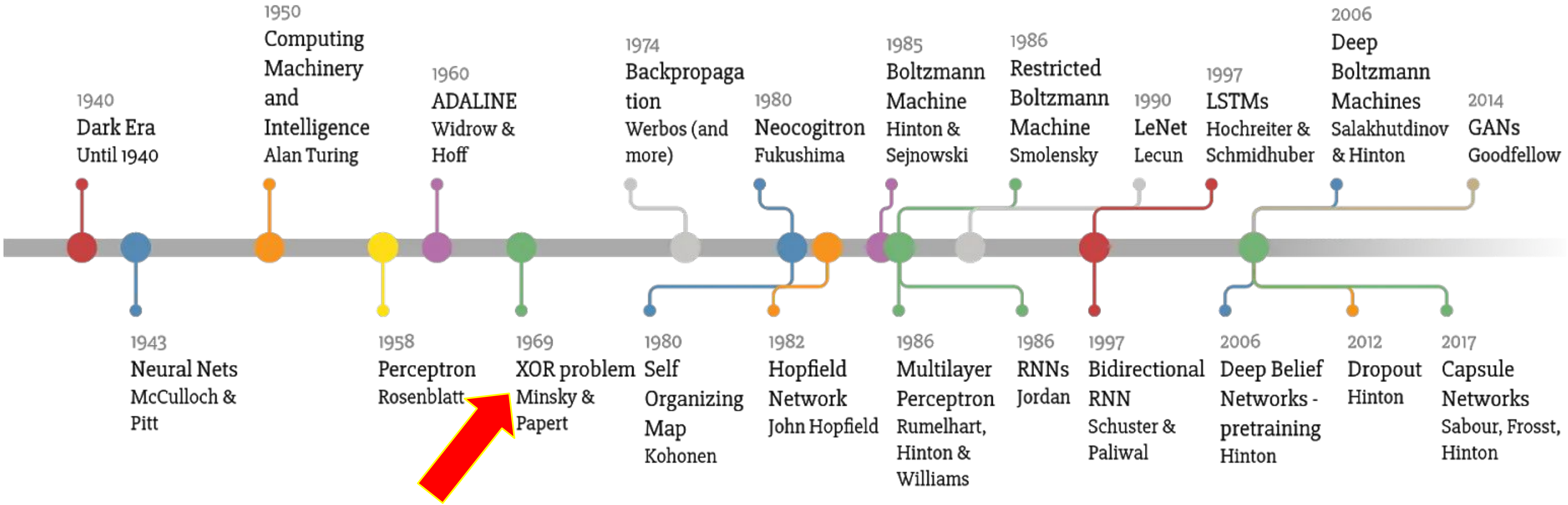
YANIKA KONGSOROT

Outline

- Limitation of Single-Layer Feedforward Neural Network
- Map the original space to the new space
- Multi-Layer Perceptron (MLP)
- Feed-Forward learning
- Weight adjusting
- MLP Backpropagation Learning Algorithm

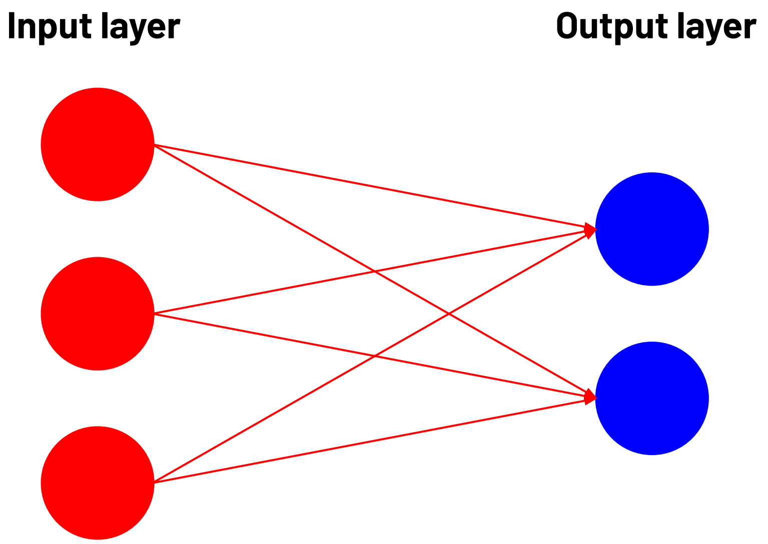
Limitation of Single-Layer Feedforward Neural Network

Deep Learning Timeline

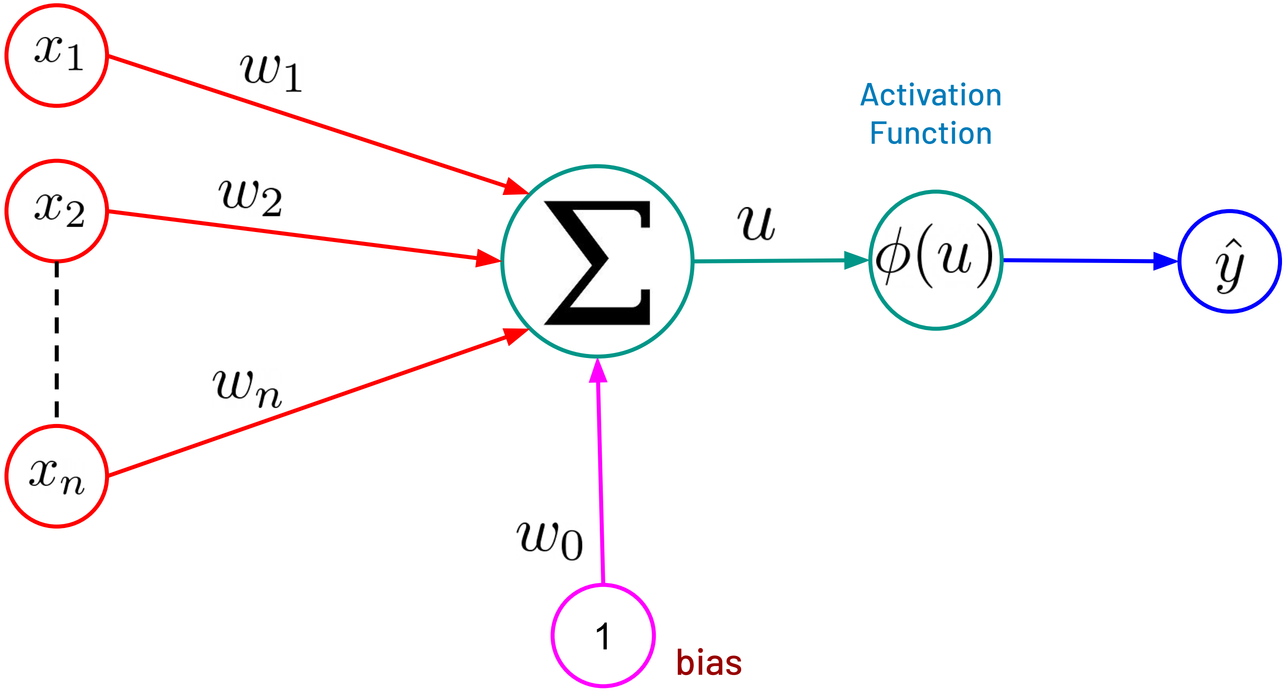


Made by Favio Vázquez

Single-layer Feedforward Neural Network



Single-Layer **Feedforward Neural Network**



Single-Layer Feedforward Neural Network

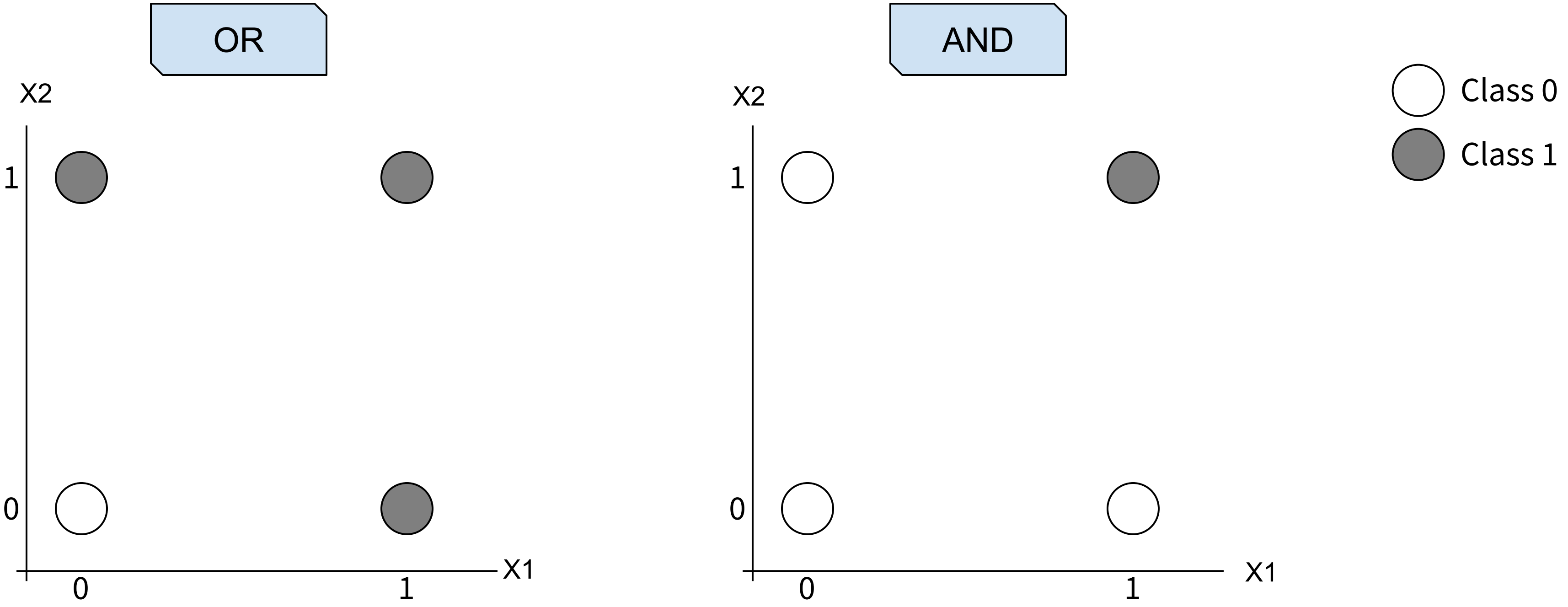
OR function

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

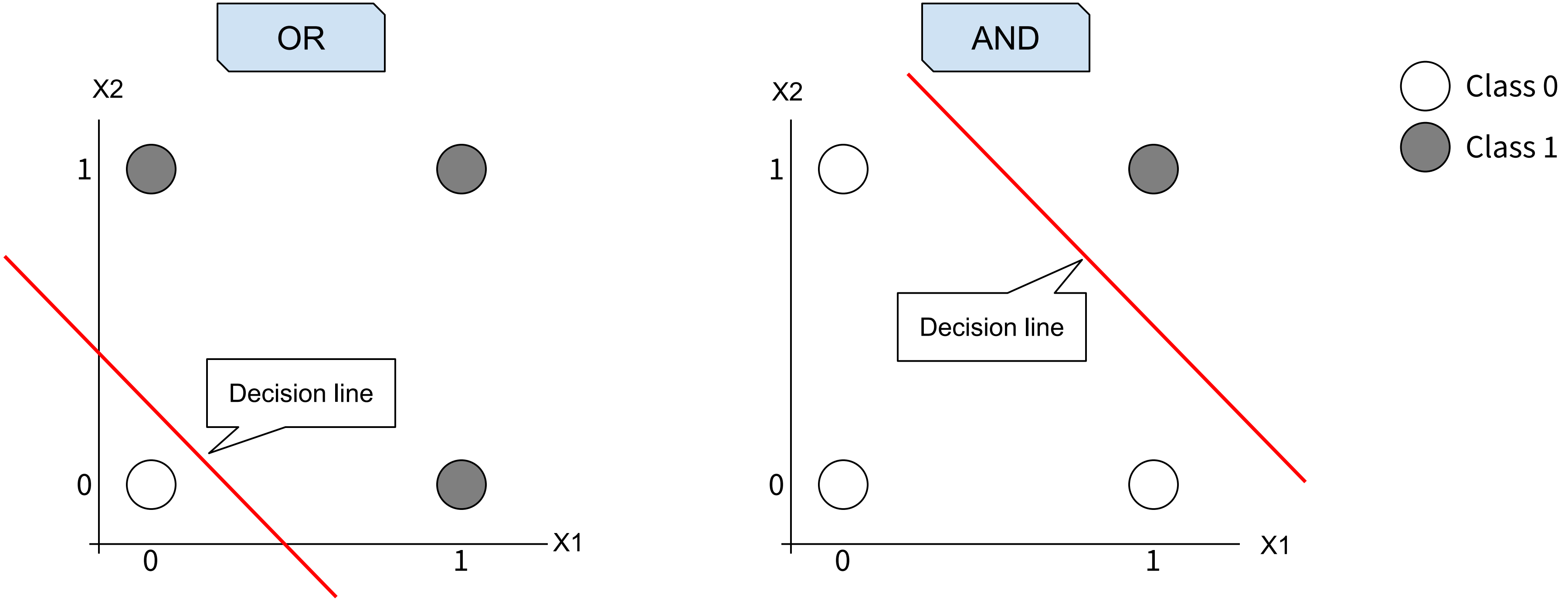
AND function

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

Single-Layer Feedforward Neural Network



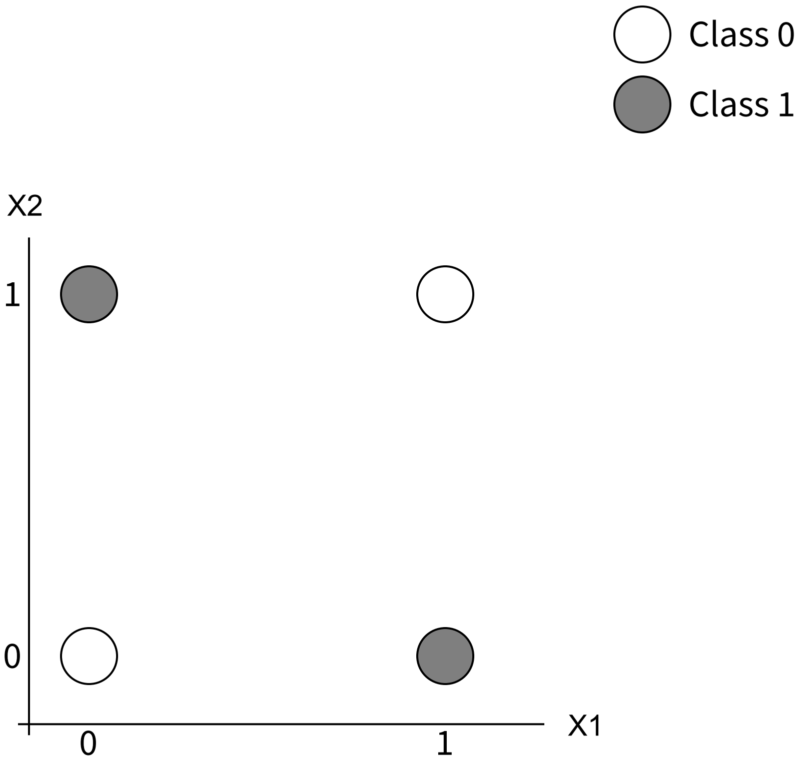
Single-Layer Feedforward Neural Network



Limitation of Single-Layer Feedforward Neural Network

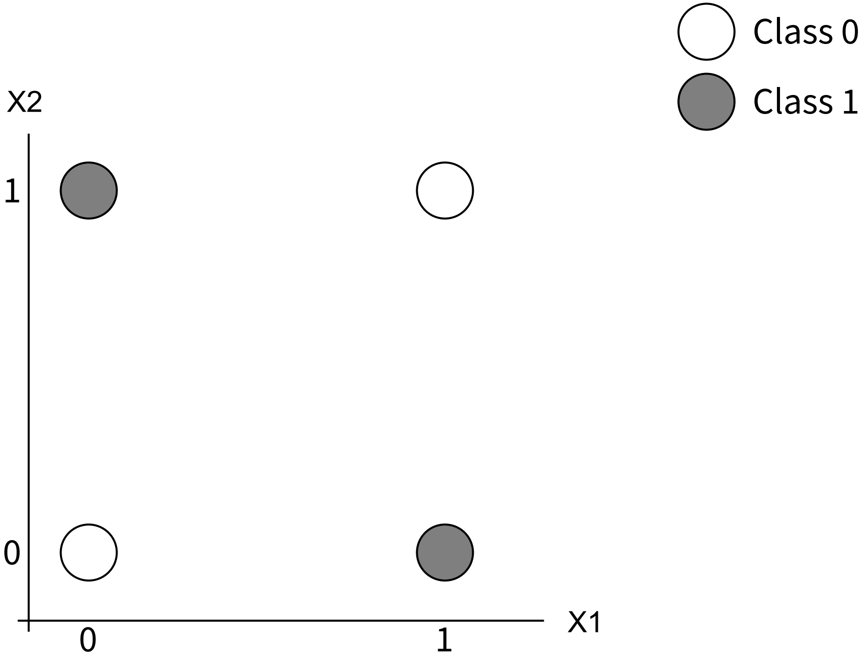
XOR function

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

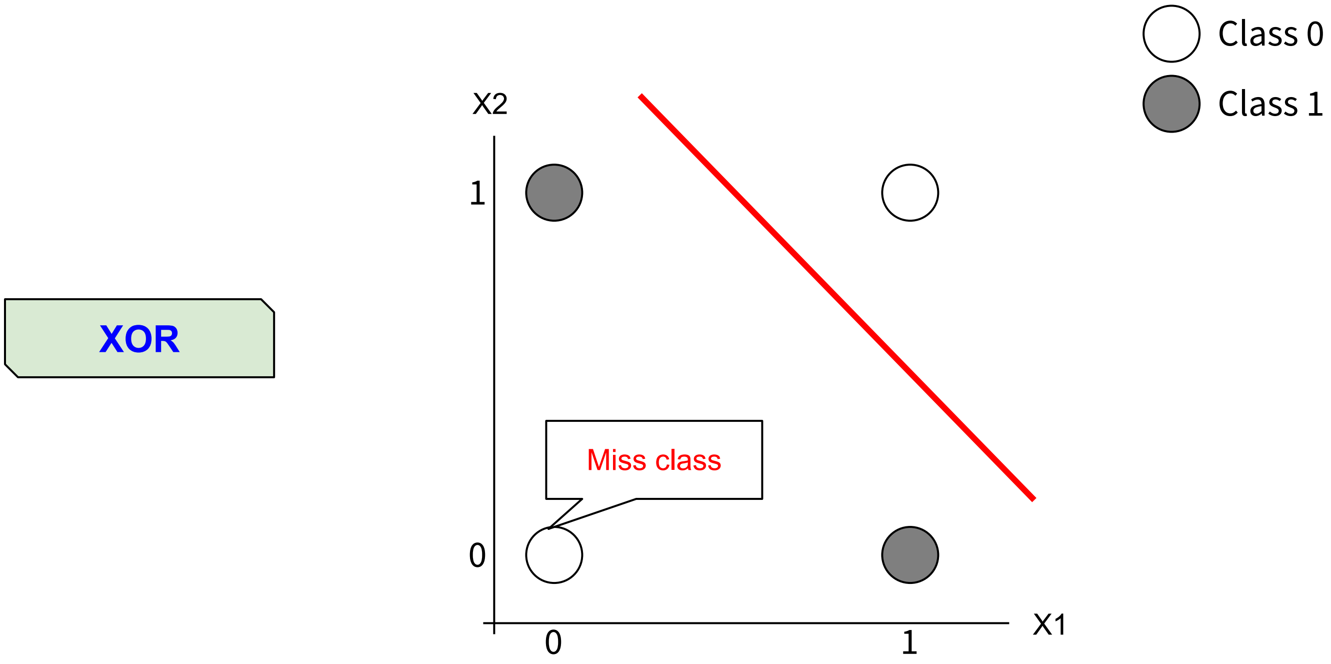


Limitation of Single-Layer Feedforward Neural Network

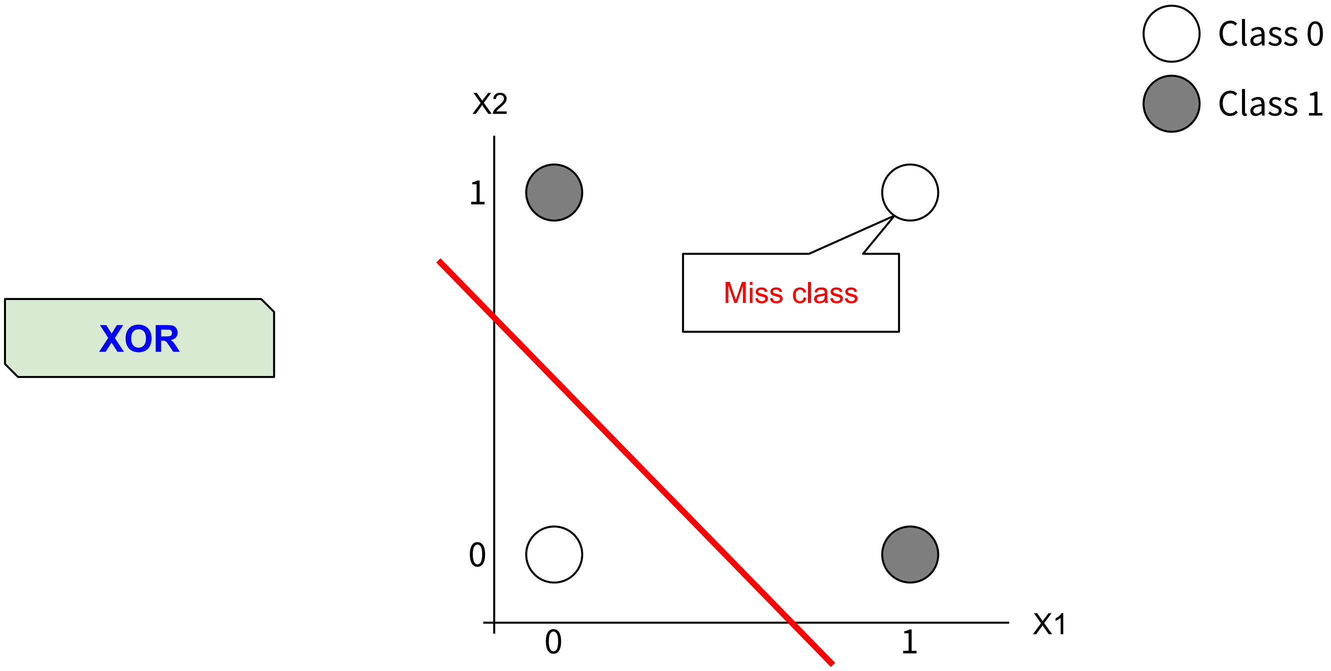
XOR



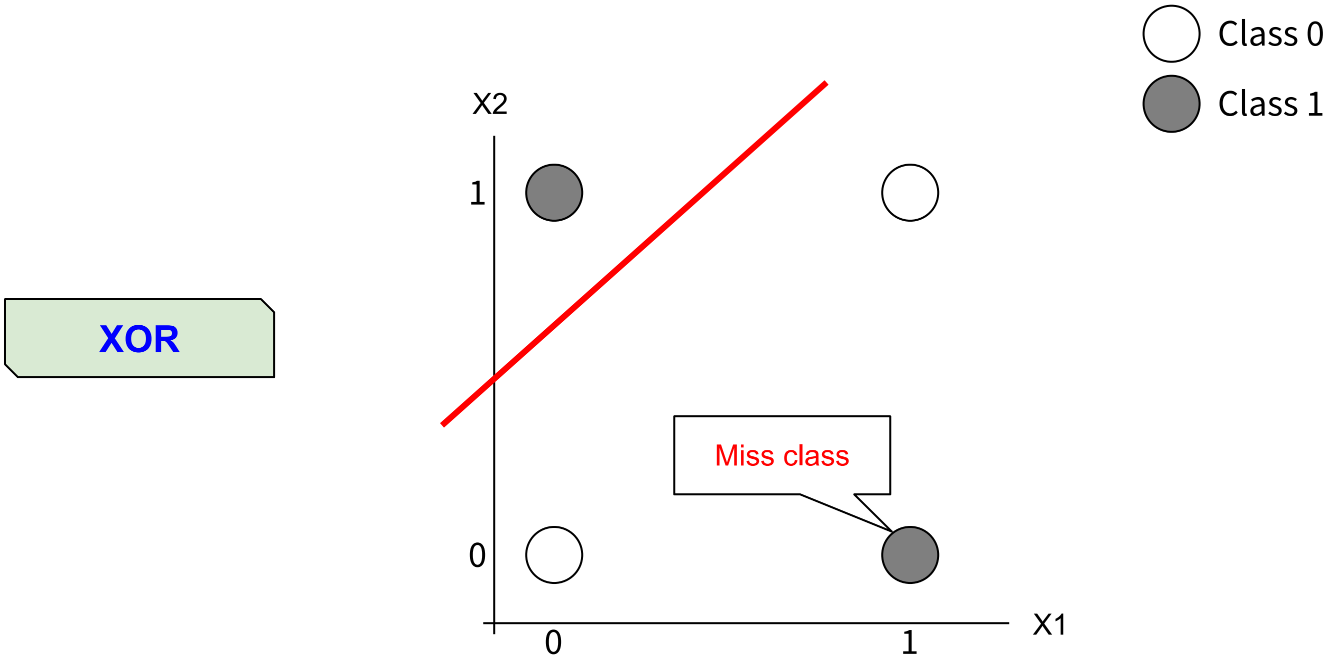
Limitation of Single-Layer Feedforward Neural Network



Limitation of Single-Layer Feedforward Neural Network

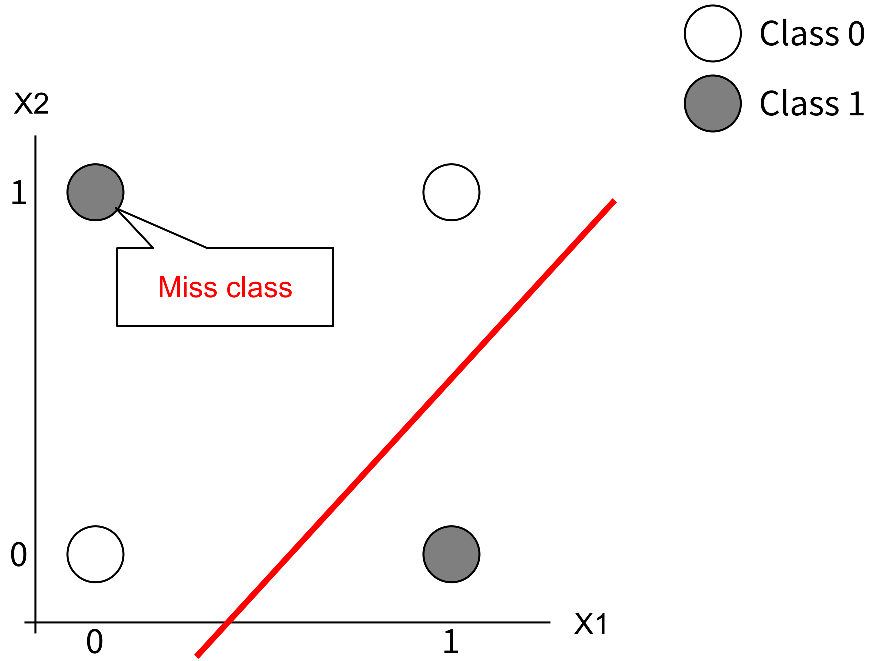


Limitation of Single-Layer Feedforward Neural Network



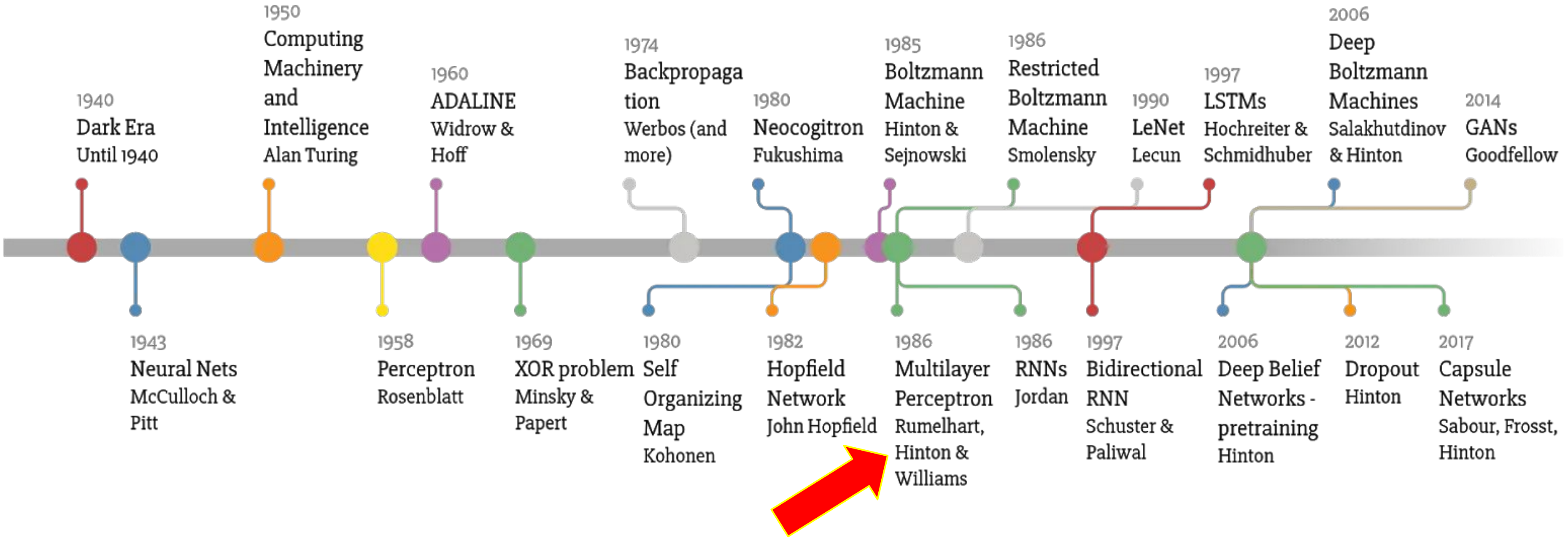
Limitation of Single-Layer Feedforward Neural Network

XOR



What can we do?

Deep Learning Timeline

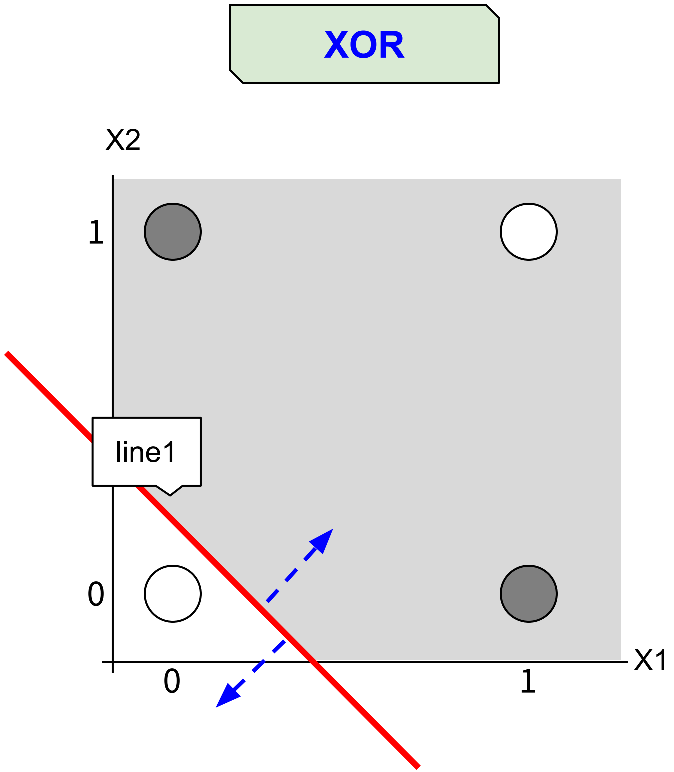


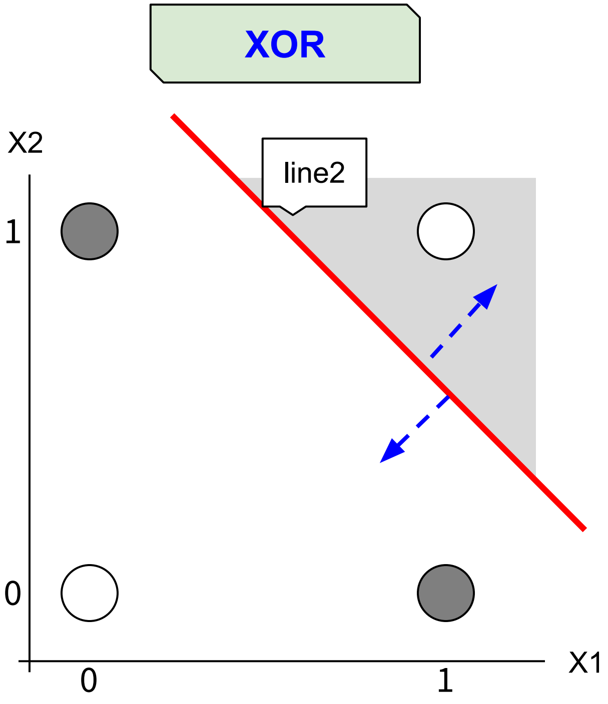
Made by Favio Vázquez

Map the original space to the new space

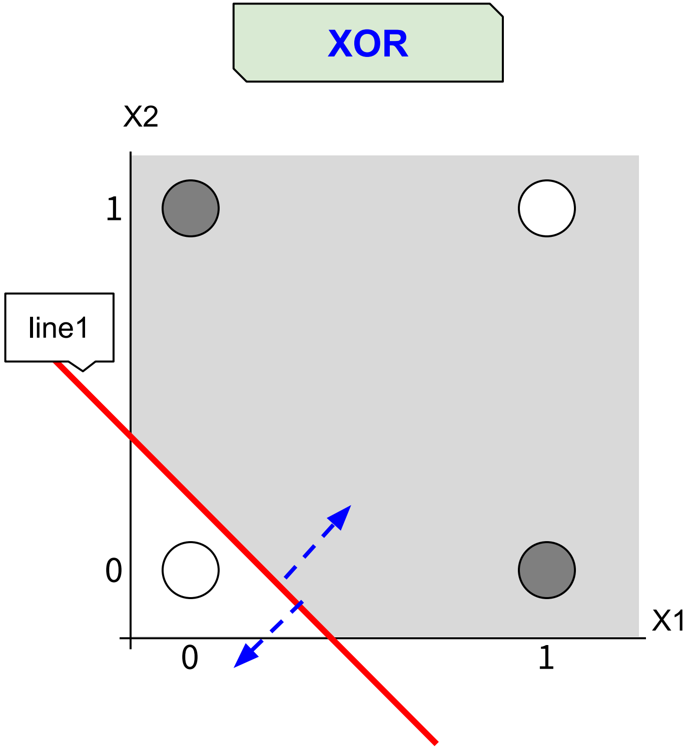
$(x1,x2)$ to $(h1,h2)$

Map the original space to the new space



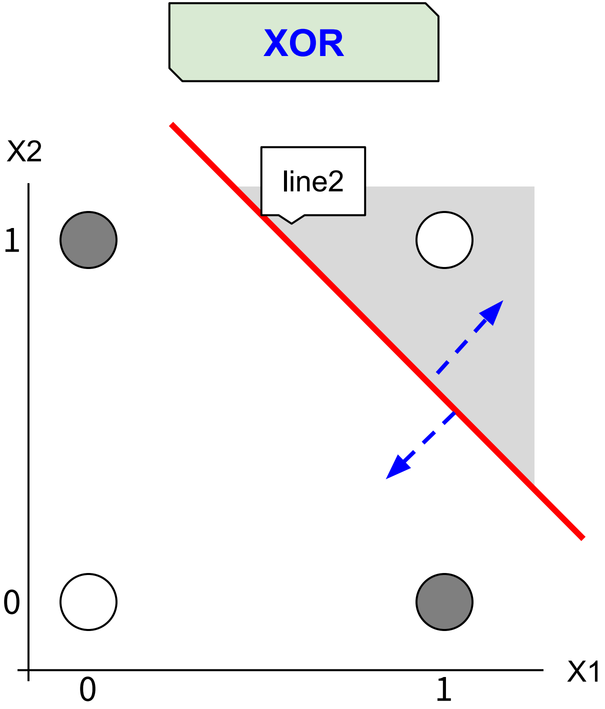


Map the original space to the new space



x_1	x_2	h_1
0	0	0
0	1	1
1	0	1
1	1	1

Map the original space to the new space

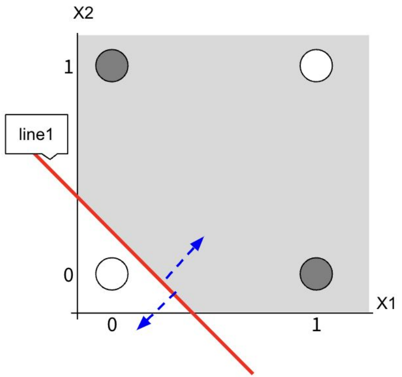


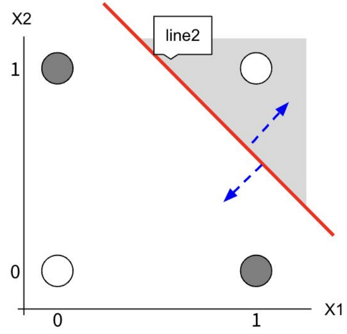
x_1	x_2	h_2
0	0	**0**
0	1	**0**
1	0	**0**
1	1	**1**

Map the original space (x_1,x_2) to the new space (h_1,h_2)

x_1	x_2	h_1
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**1**

x_1	x_2	h_2
0	0	**0**
0	1	**0**
1	0	**0**
1	1	**1**



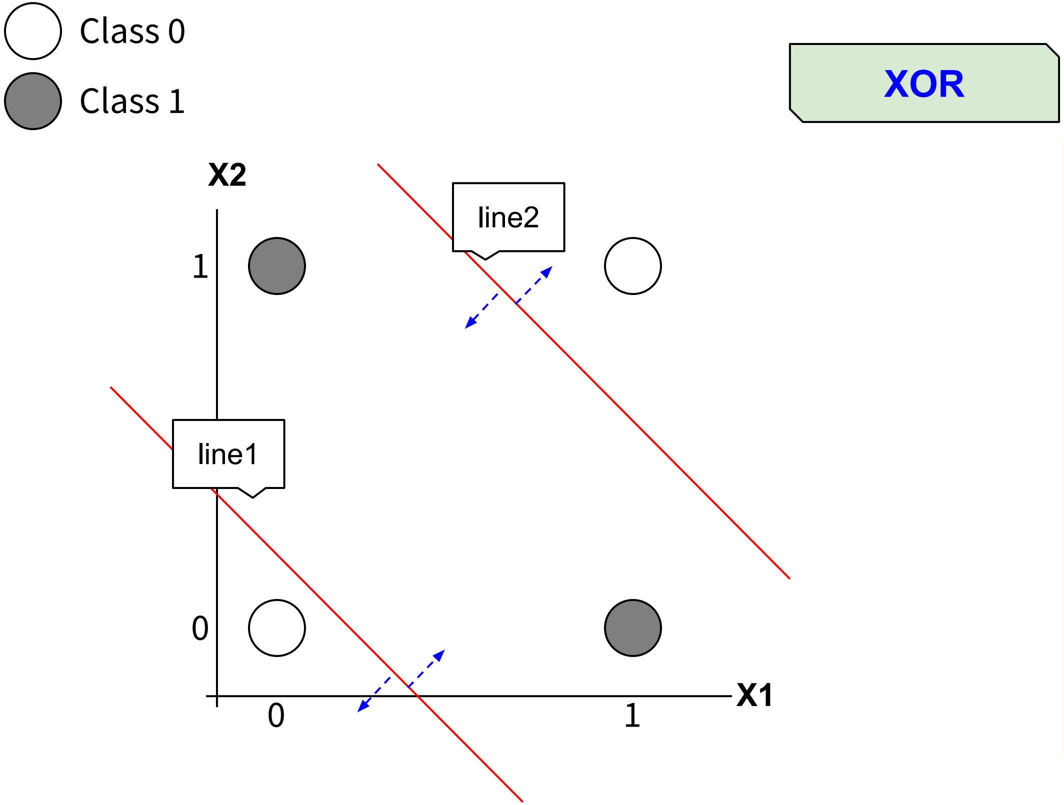


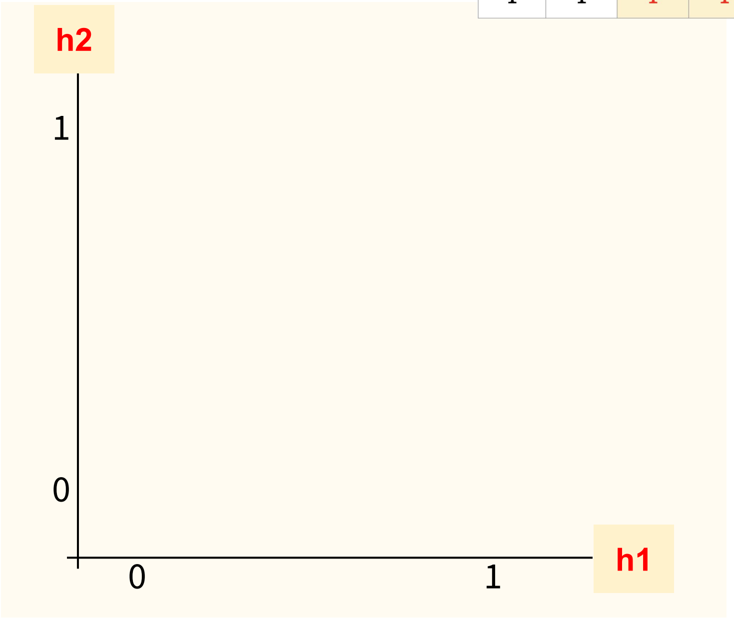
Map the original space (x_1, x_2) to the new space (h_1, h_2)

x_1	x_2	h_1	h_2	y
0	0	0	0	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	0

Map the original space (x_1, x_2) to the new space (h_1, h_2)

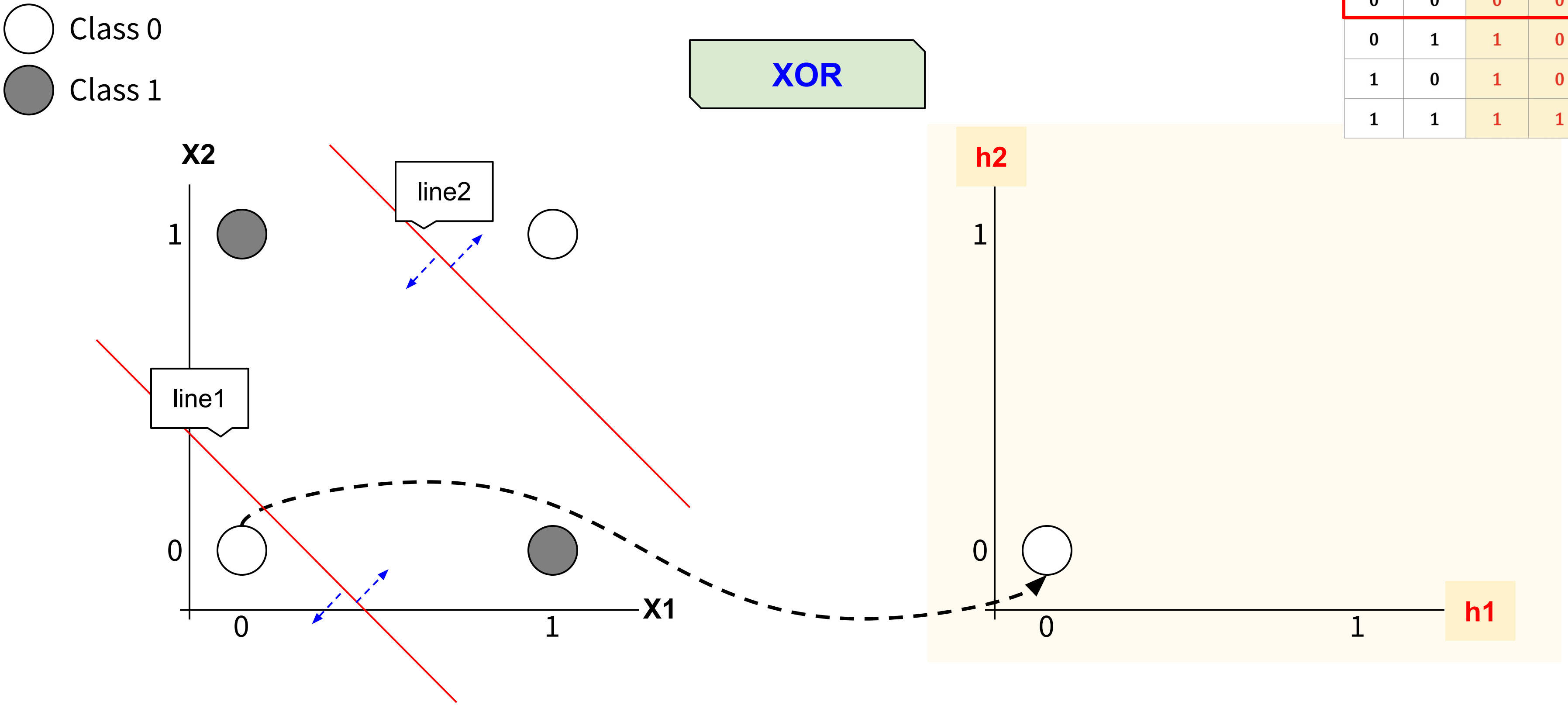
x_1	x_2	h_1	h_2	y
0	0	0	0	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	0





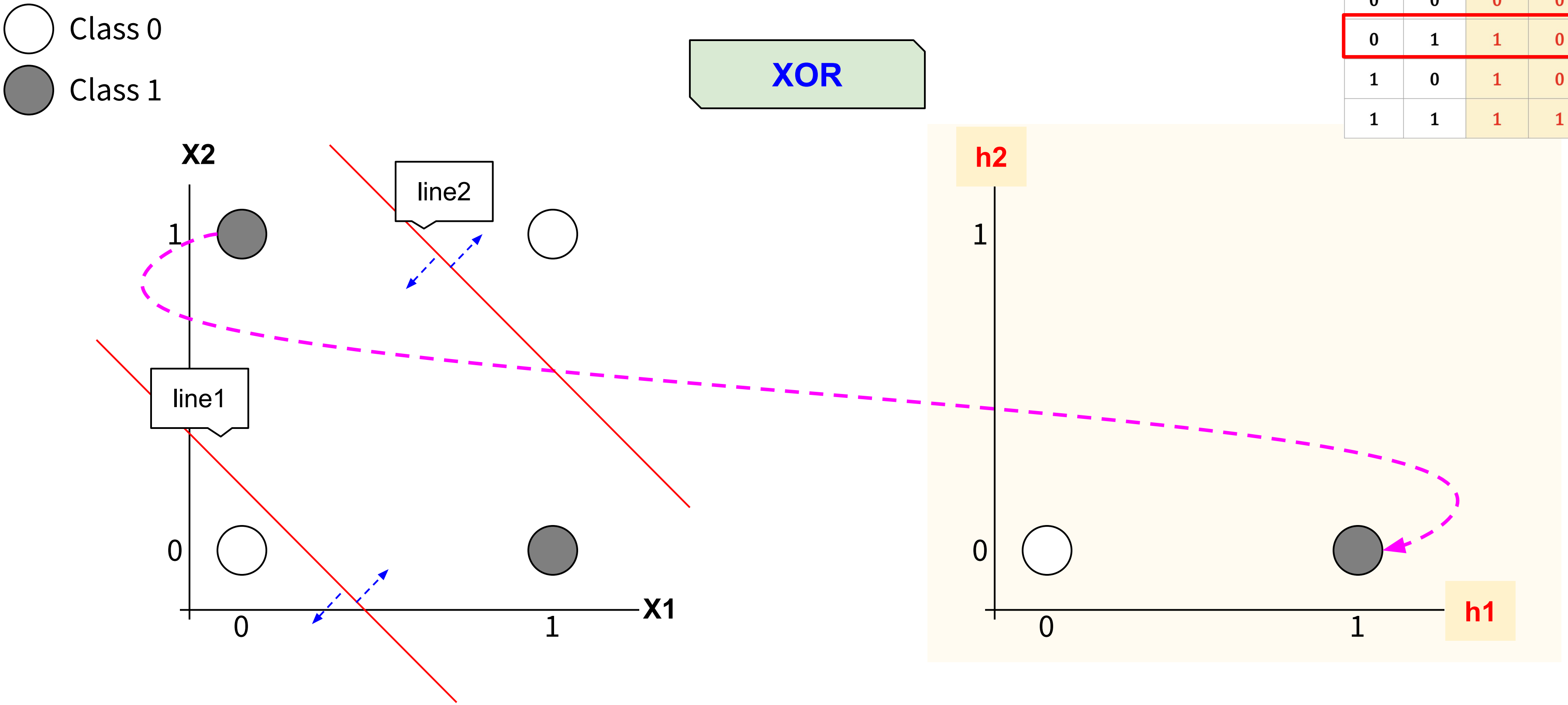
Map the original space (x_1, x_2) to the new space (h_1, h_2)

x_1	x_2	h_1	h_2	y
0	0	0	0	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	0



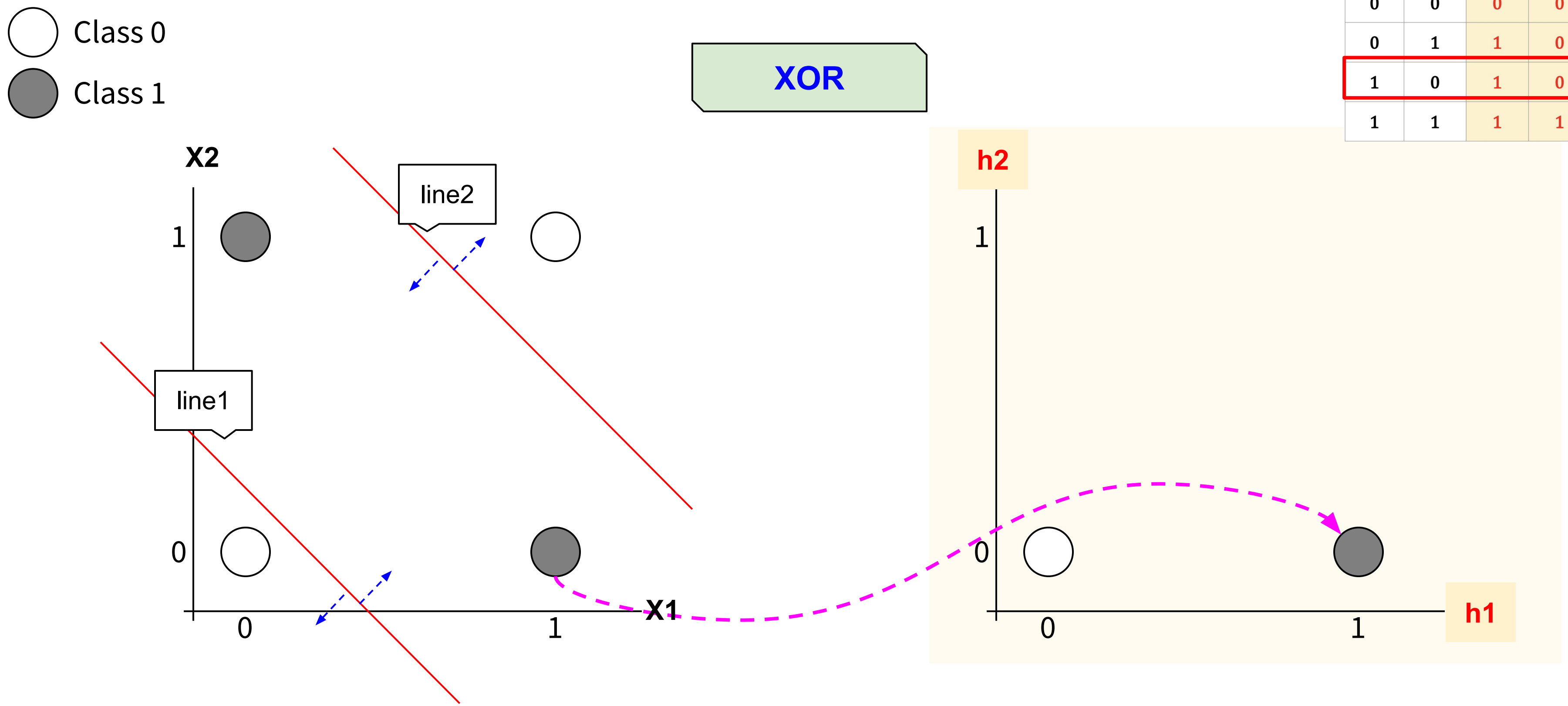
Map the original space (x_1, x_2) to the new space (h_1, h_2)

x_1	x_2	h_1	h_2	y
0	0	0	0	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	0



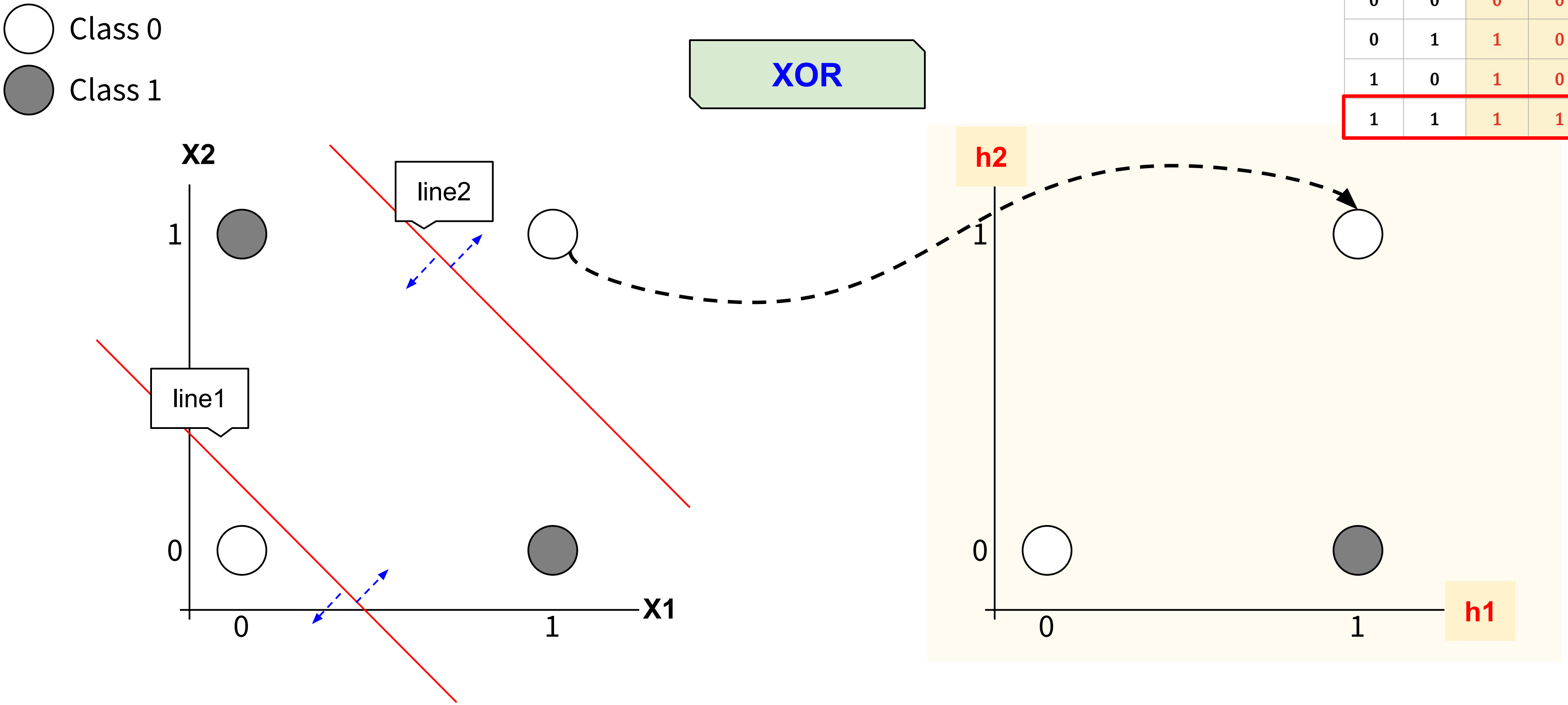
Map the original space (x_1,x_2) to the new space (h_1,h_2)

x_1	x_2	h_1	h_2	y
0	0	0	0	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	0



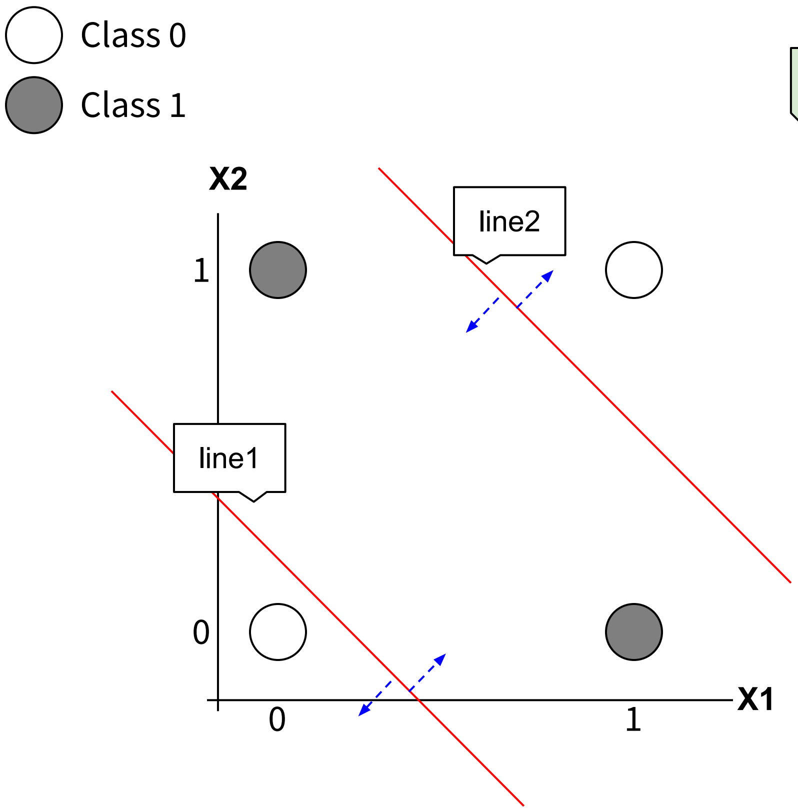
Map the original space (x_1, x_2) to the new space (h_1, h_2)

x_1	x_2	h_1	h_2	y
0	0	0	0	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	0

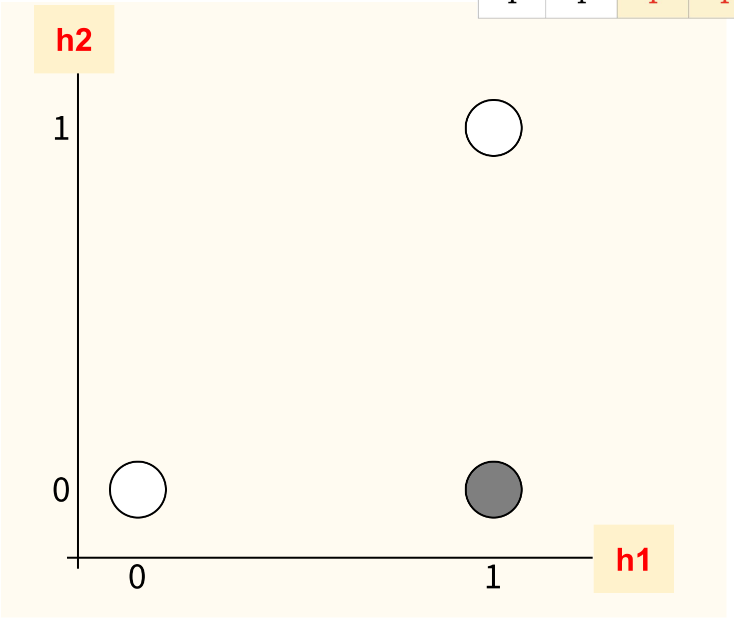


Map the original space (x_1, x_2) to the new space (h_1, h_2)

x_1	x_2	h_1	h_2	y
0	0	0	0	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	0



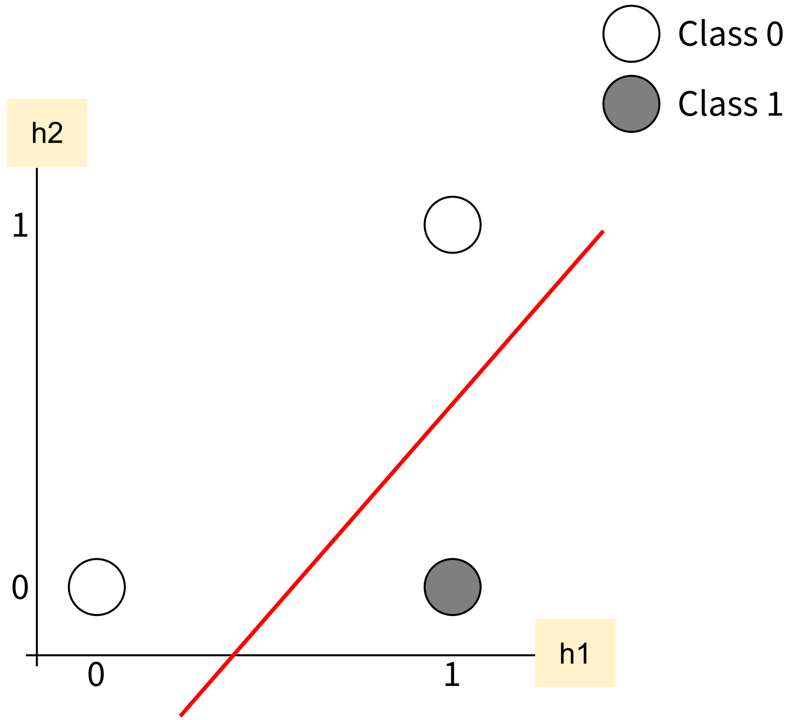
XOR



Map the original space (x_1,x_2) to the new space (h_1,h_2)

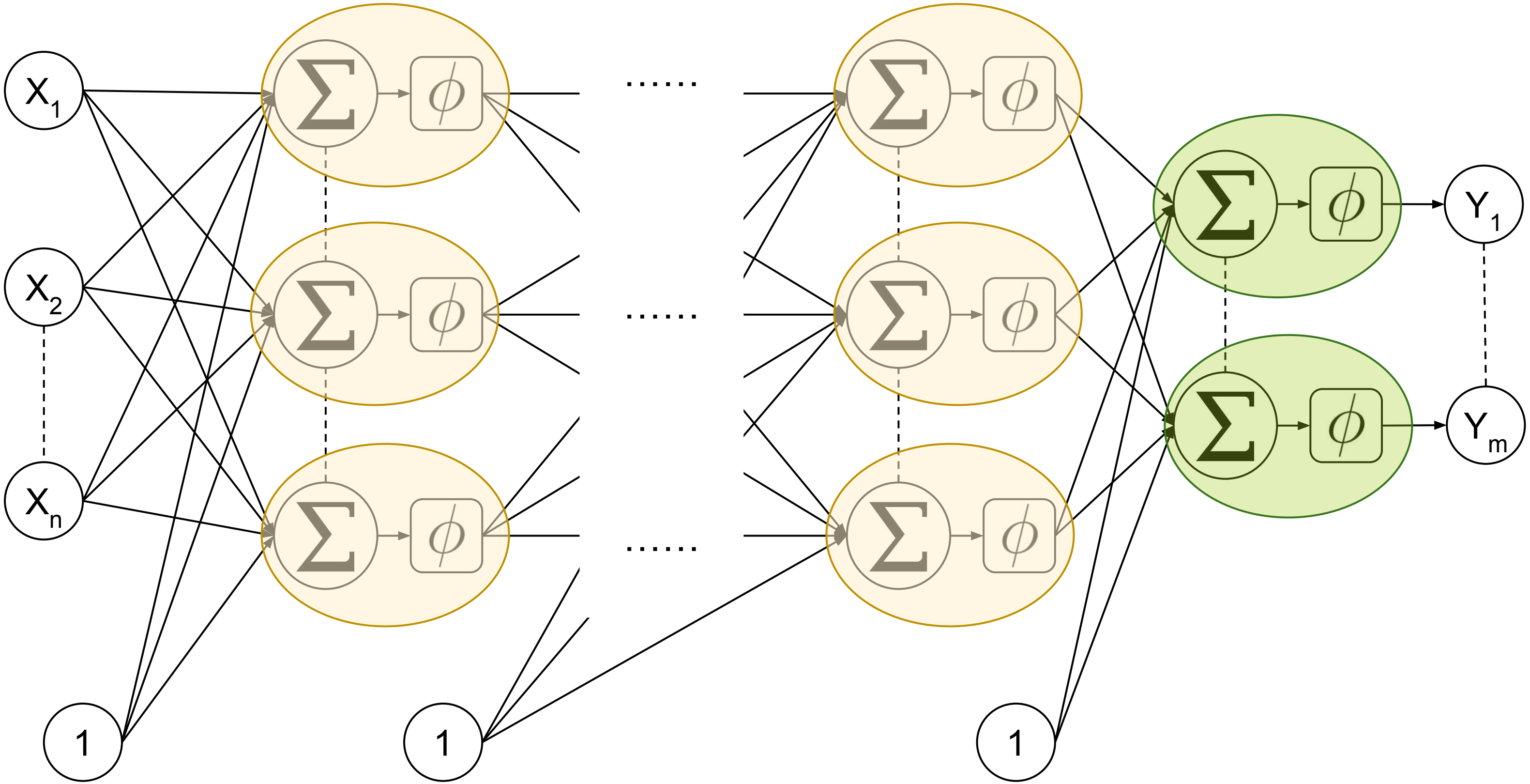
XOR

x_1	x_2	h_1	h_2	y
0	0	0	0	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	0

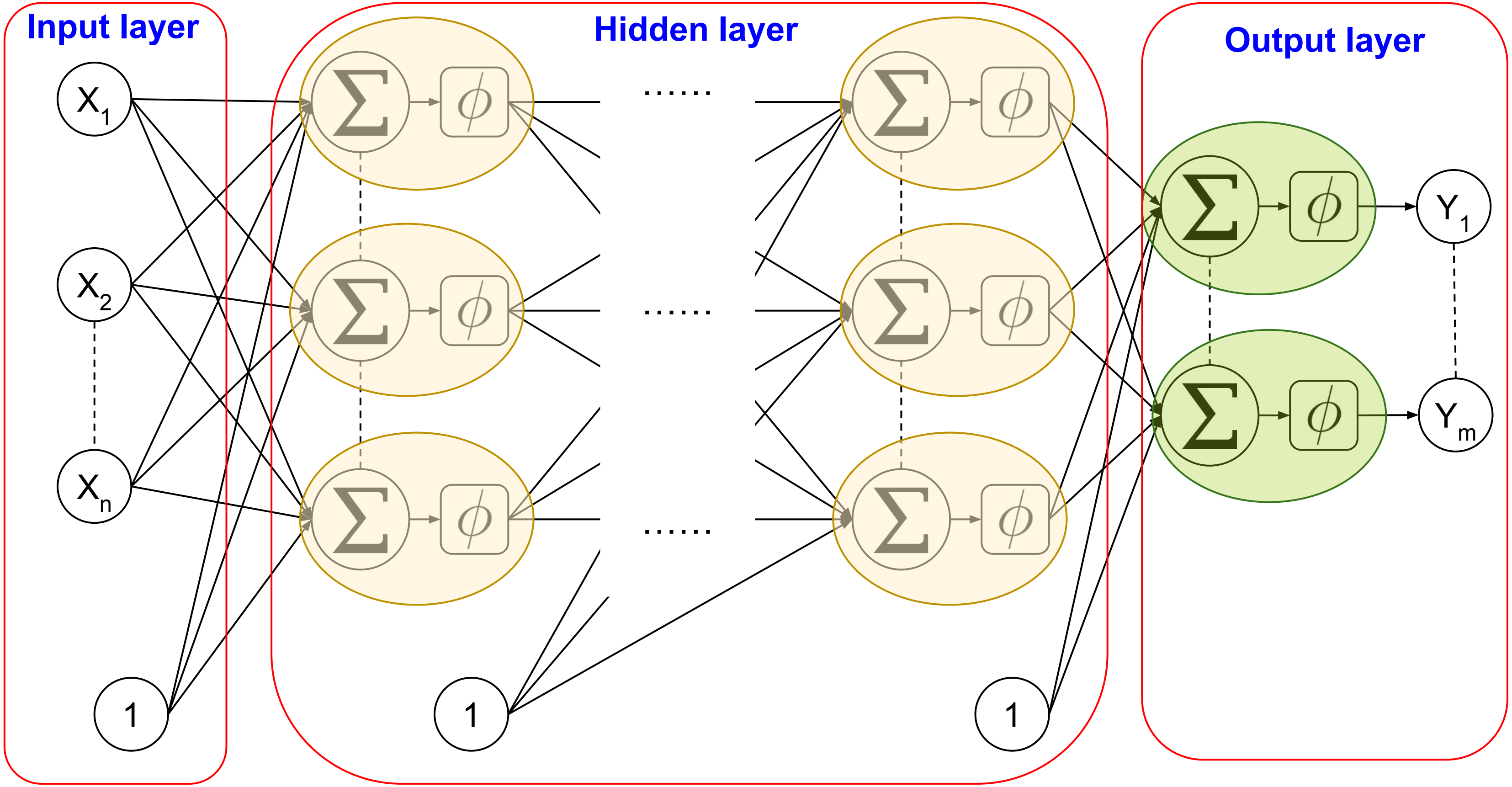


Multi-Layer Perceptron (MLP)

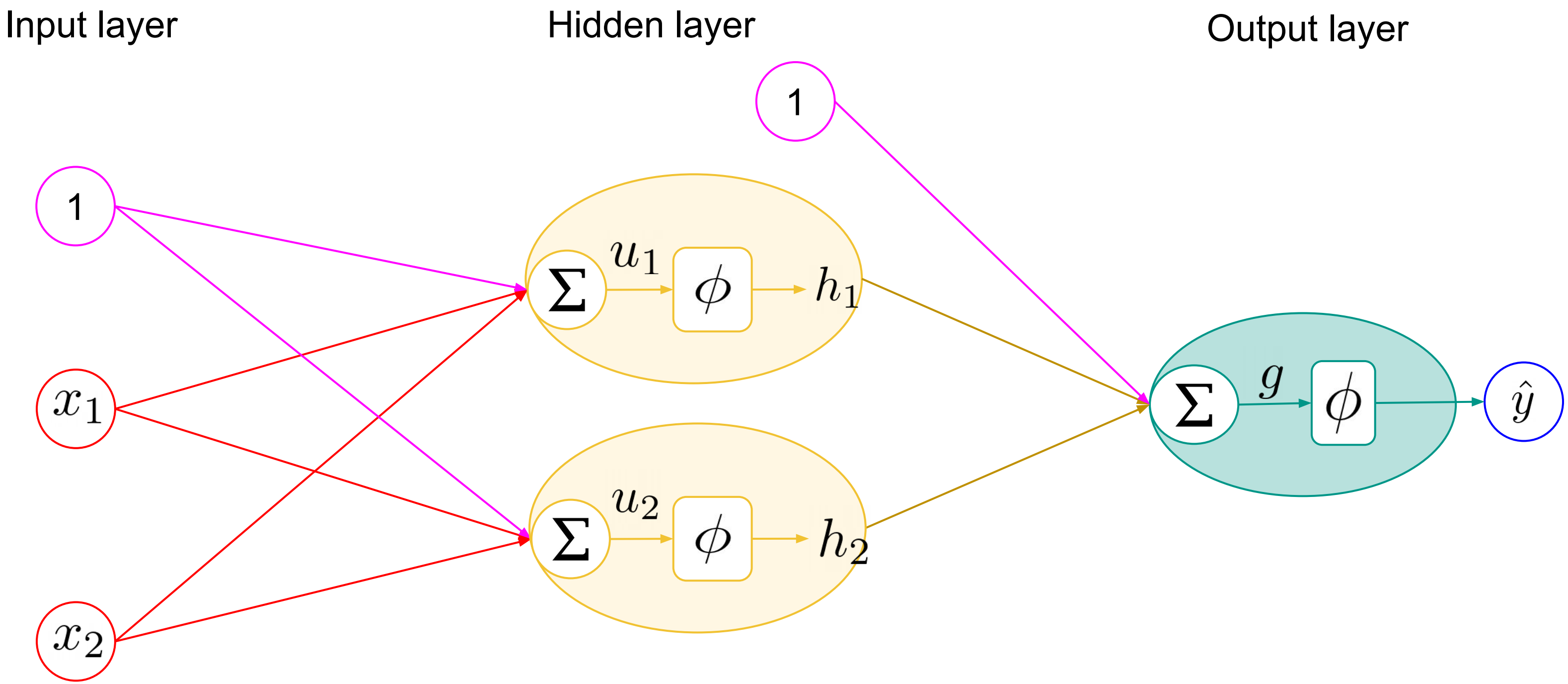
Multi-Layer Perceptron (MLP)



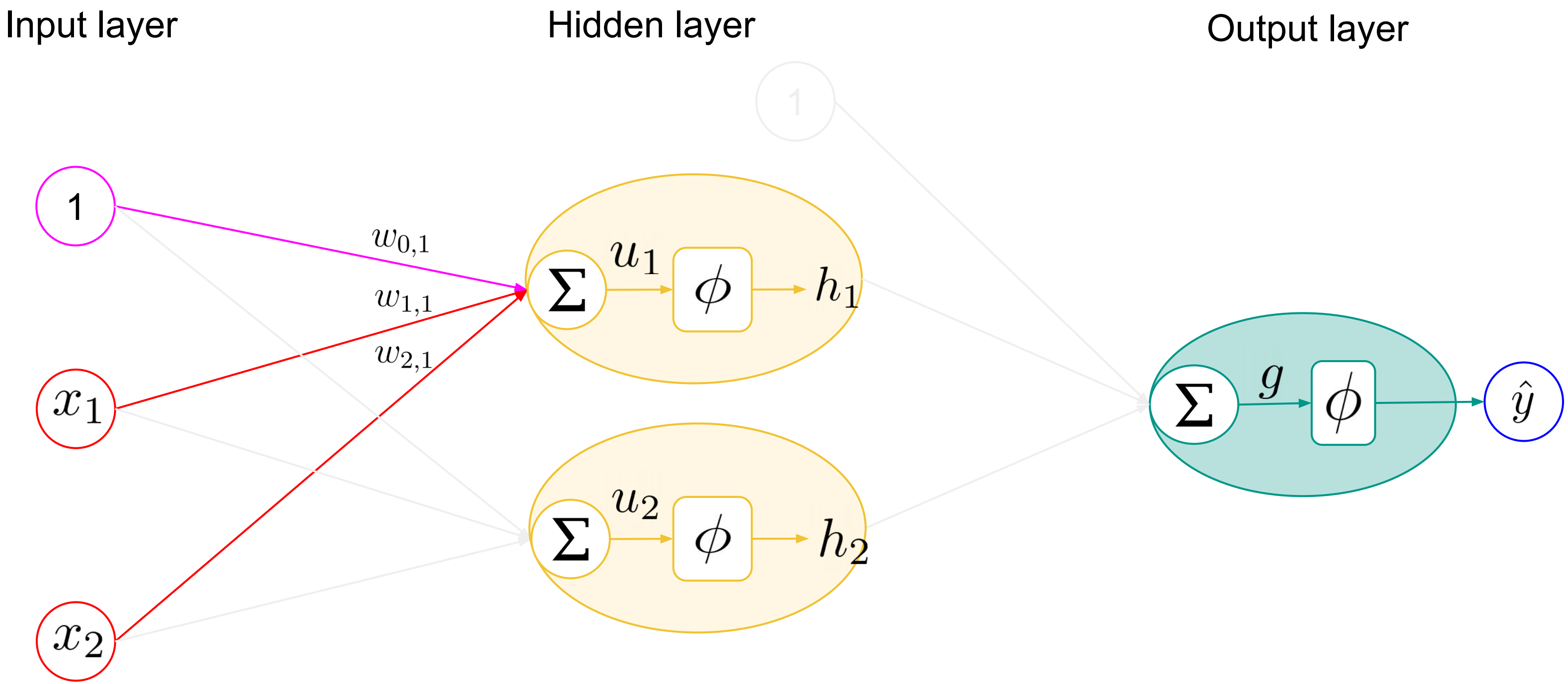
Multi-Layer Perceptron (MLP)



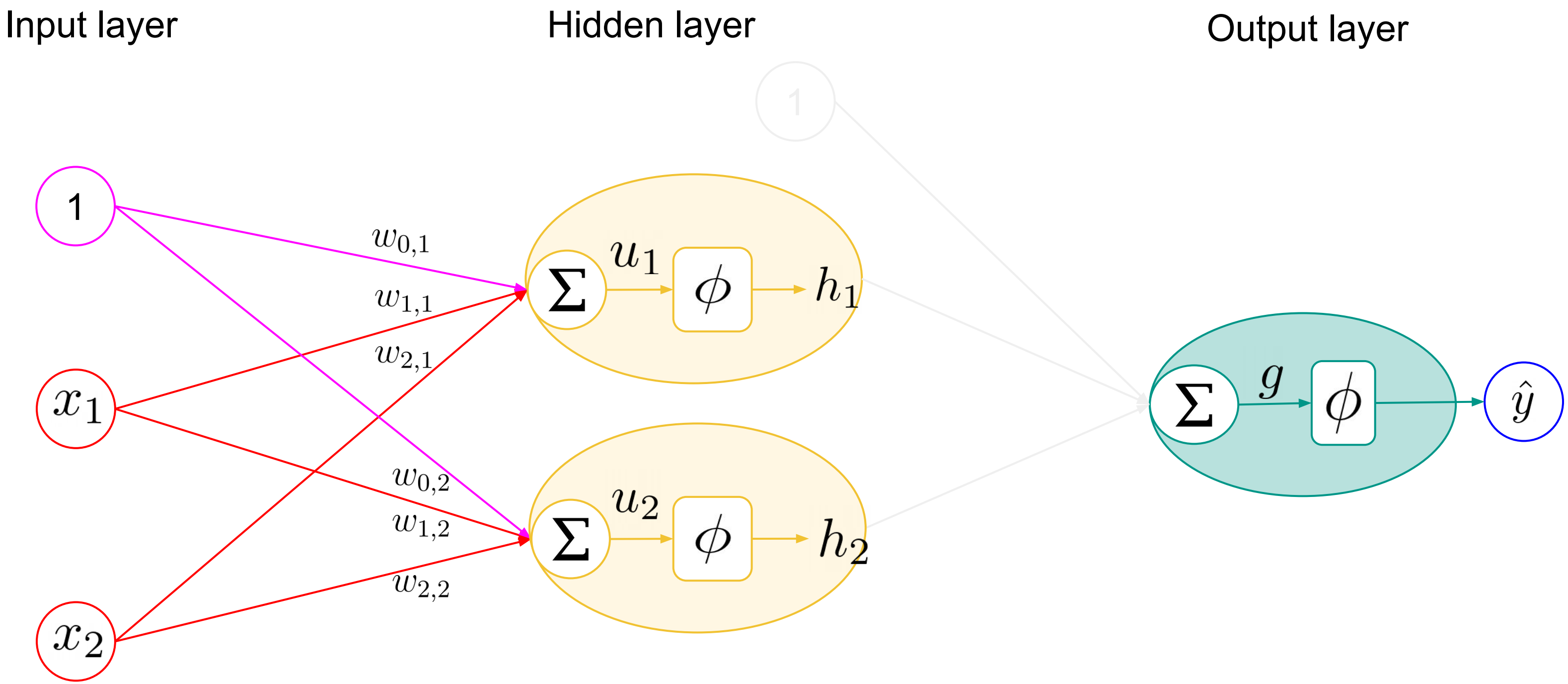
Multi-Layer Perceptron (MLP)



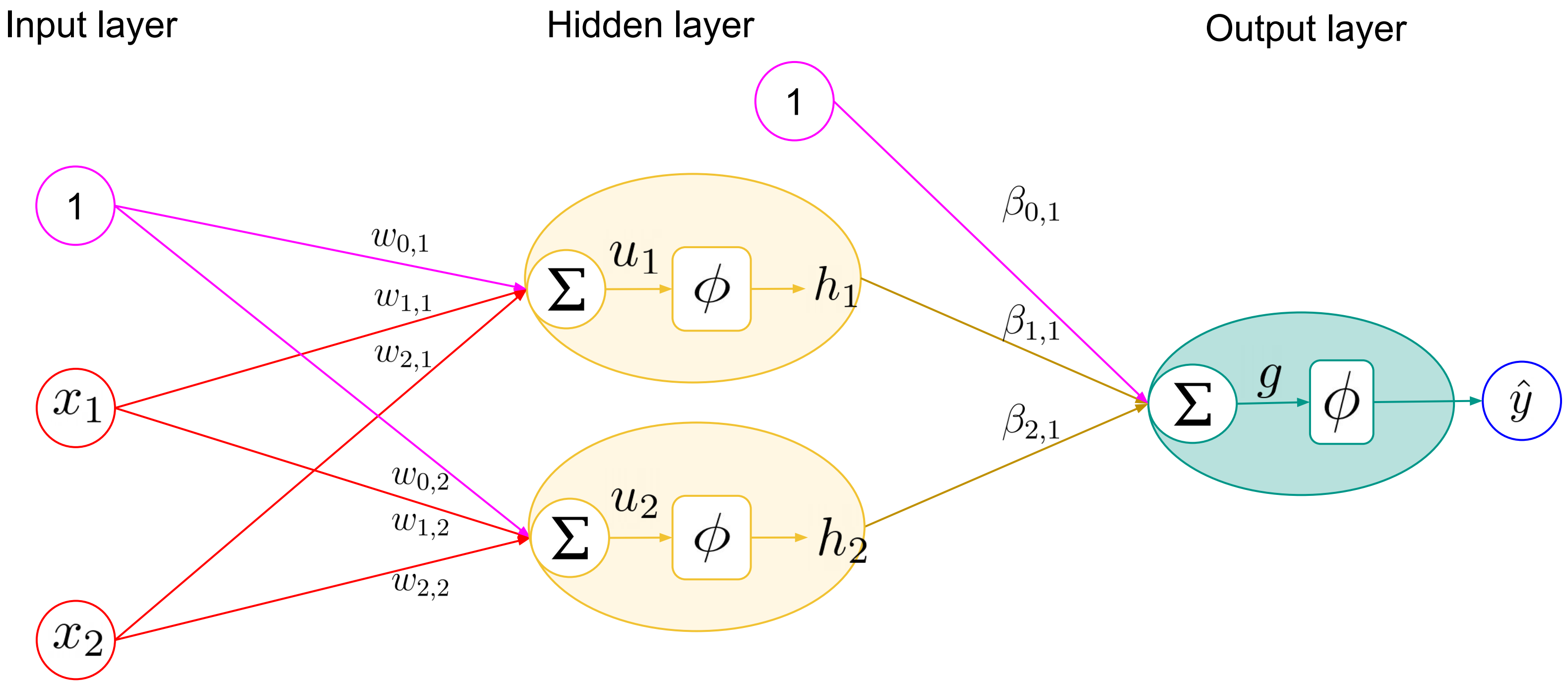
Multi-Layer Perceptron (MLP)



Multi-Layer Perceptron (MLP)



Multi-Layer Perceptron (MLP)



How to train MLP?

How to train MLP?

Backpropagation Learning Algorithm

Feed-Forward

- Compute output

Backpropagation

- Weight tuning

How to train MLP?

Step 1: parameter setting

- Define network architecture
  - number of hidden layers
  - number of hidden nodes
- Set the initial weights
- Define Activation function
- Define the value of learning rate
- Define the stopping criteria
  - (i.e.) number of round

How to train MLP?

Step 2: Train Model by *Backpropagation Learning Algorithm*

- For each data point ($\mathbf{x}$)

  Feed-Forward

  - **Step 2.1:** Compute *outputs of hidden layer (h)*
  - **Step 2.2:** Compute *outputs of output layer (y_hat)*

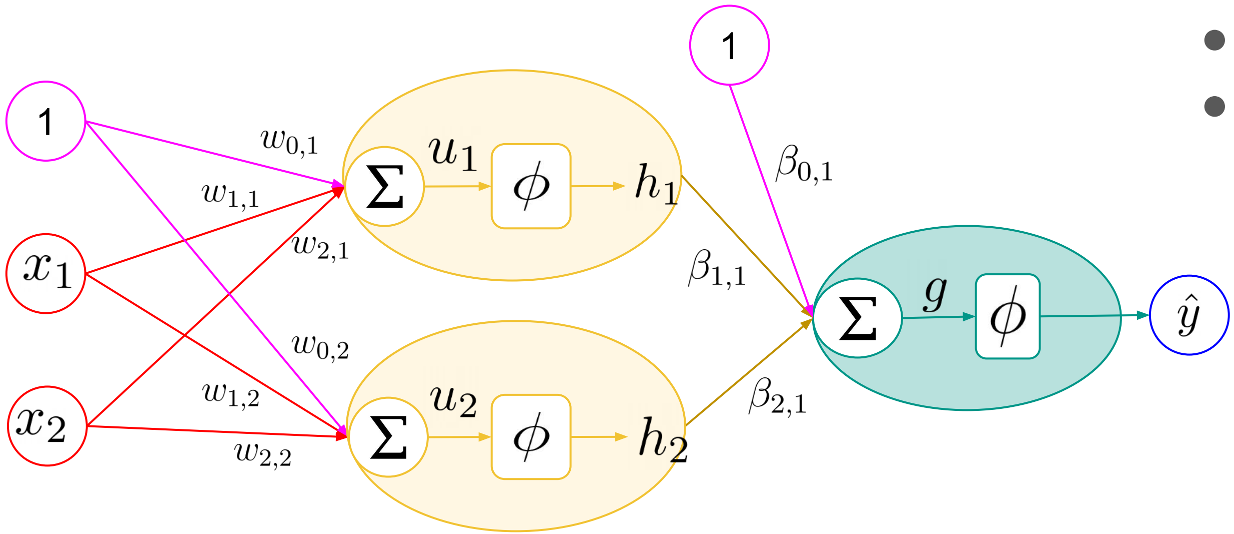
  Backpropagation

  - **Step 2.3:** Adjust the *weights of output layer*
  - **Step 2.4:** Adjust the *weights of input (hidden) layer*

Feed-Forward

Compute the output of network

Feed-Forward

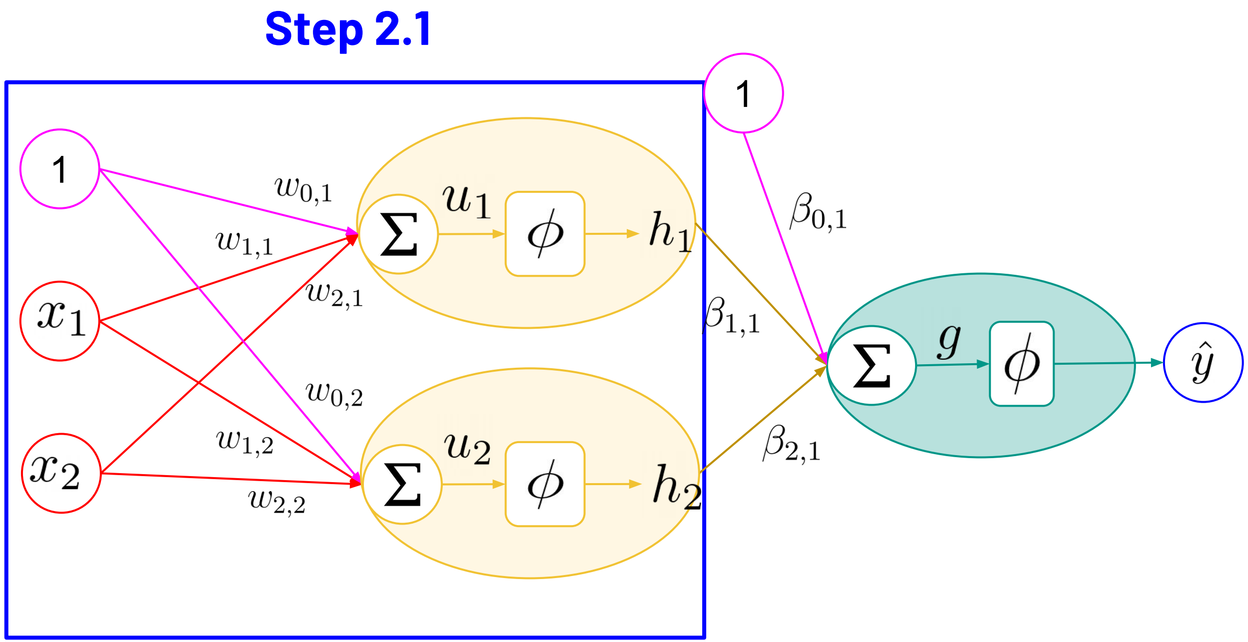


Network architecture: **[2-2-1]**

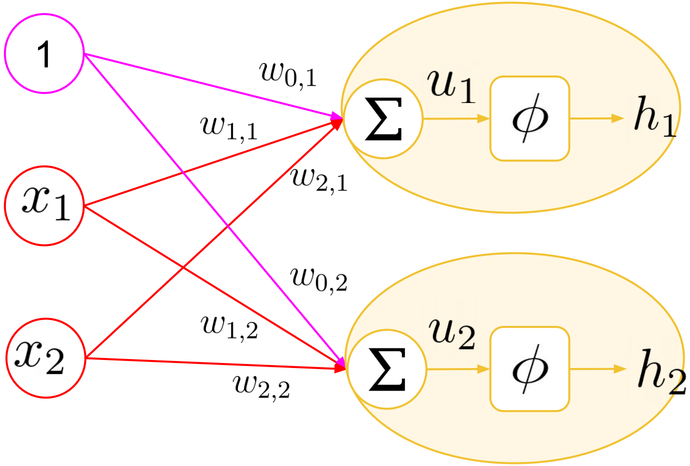
- 2 input nodes
- 1 hidden layer
- 2 hidden nodes
- 1 output node

$$\phi(u) = \frac{1}{1 + e^{-u}}$$

Feed-Forward



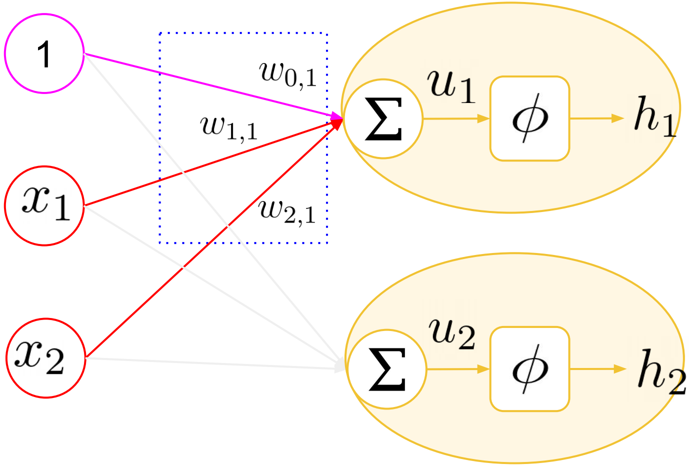
Feed-Forward



Step 2.1

$$\phi(u) = \frac{1}{1 + e^{-u}}$$

Feed-Forward

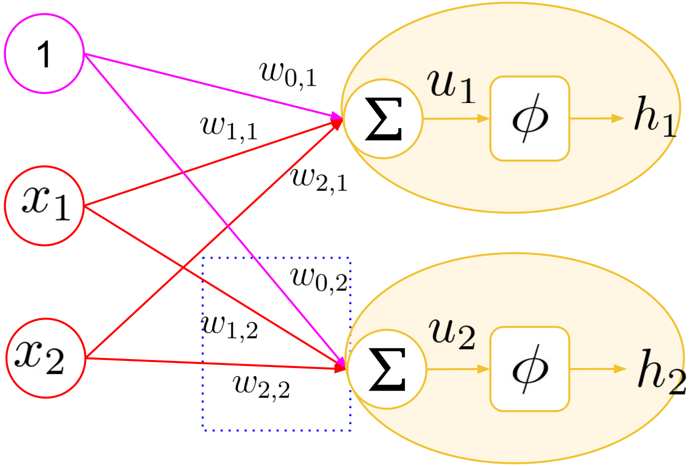


Step 2.1

$$u_1 = w_{0,1} + \sum_{j=1}^{n} w_{j,1} x_j \longrightarrow h_1 = \phi(u_1)$$

$$\phi(u) = \frac{1}{1 + e^{-u}}$$

Feed-Forward



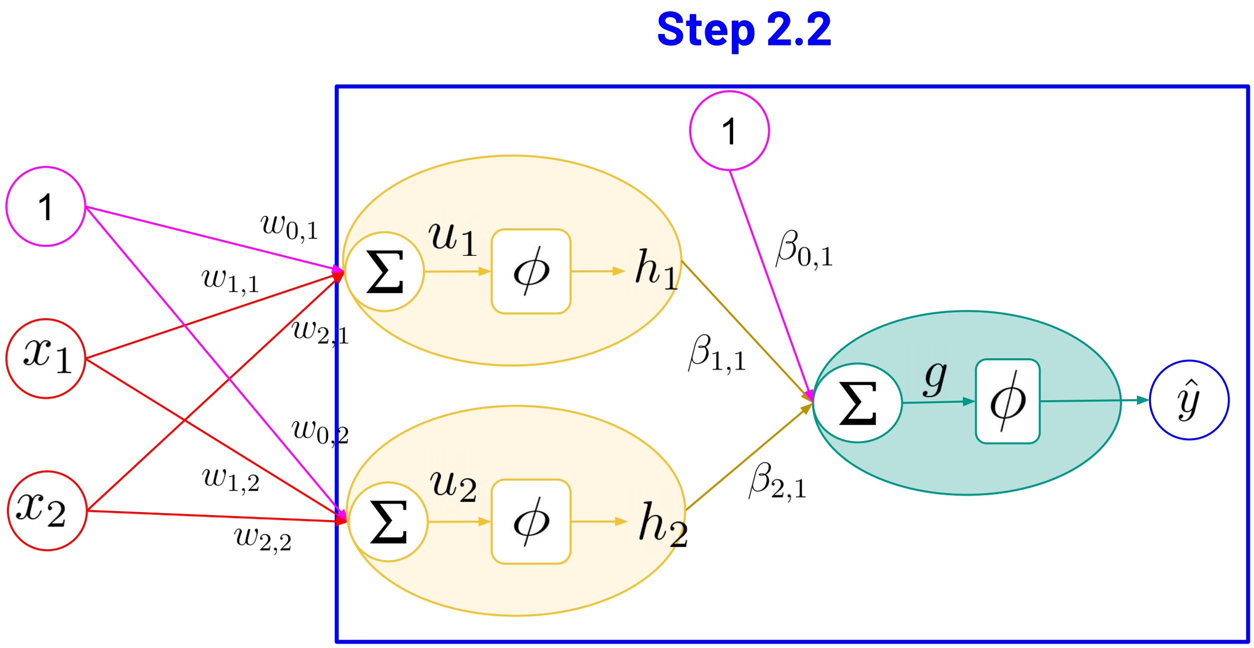
Step 2.1

$$u_1 = w_{0,1} + \sum_{j=1}^{n} w_{j,1} x_j \longrightarrow h_1 = \phi(u_1)$$

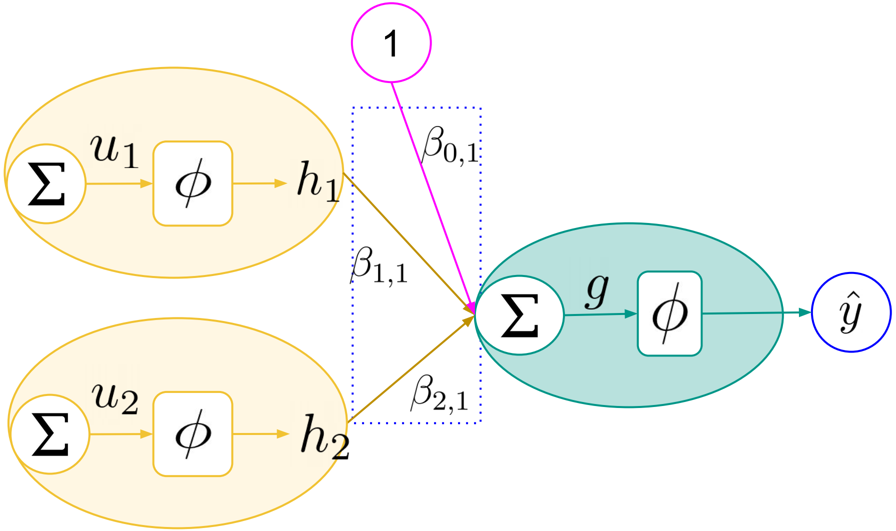
$$u_2 = w_{0,2} + \sum_{j=1}^{n} w_{j,2} x_j \longrightarrow h_2 = \phi(u_2)$$

$$\phi(u) = \frac{1}{1 + e^{-u}}$$

Feed-Forward



Feed-Forward

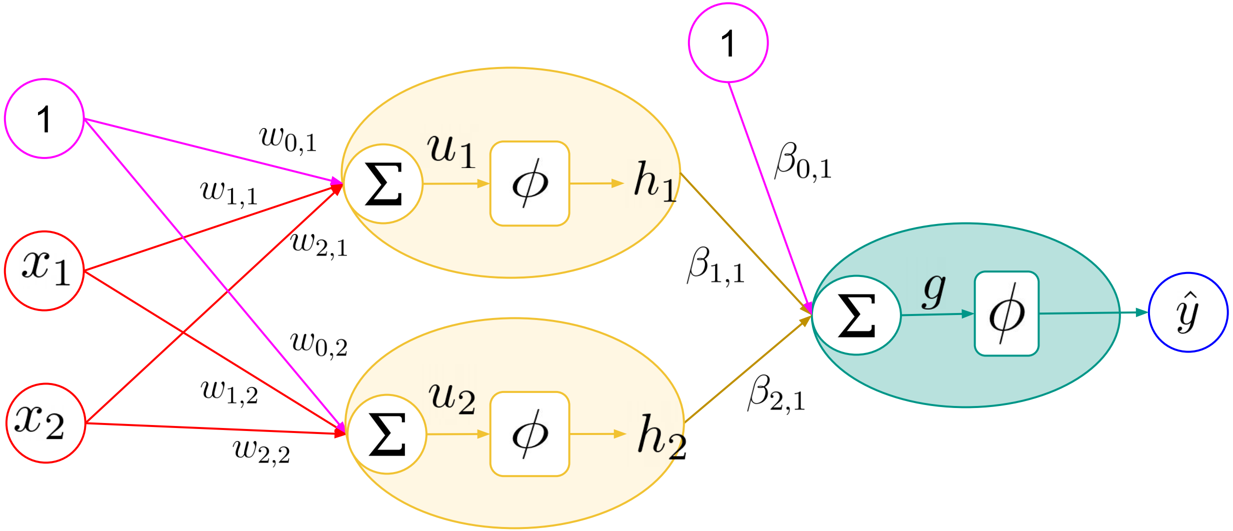


Step 2.2

$$g = \beta_{0,1} + \sum_{k=1}^{l} \beta_{k,1} h_k$$

$$\hat{y} = \phi(g)$$

Feed-Forward



Step 2.1

$$u_1 = w_{0,1} + \sum_{j=1}^{n} w_{j,1} x_j$$

$$u_2 = w_{0,2} + \sum_{j=1}^{n} w_{j,2} x_j$$

$$h_1 = \phi(u_1)$$

$$h_2 = \phi(u_2)$$

Step 2.2

$$g = \beta_{0,1} + \sum_{k=1}^{l} \beta_{k,1} h_k$$

$$\hat{y} = \phi(g)$$

Backpropagation

Weight adjustment

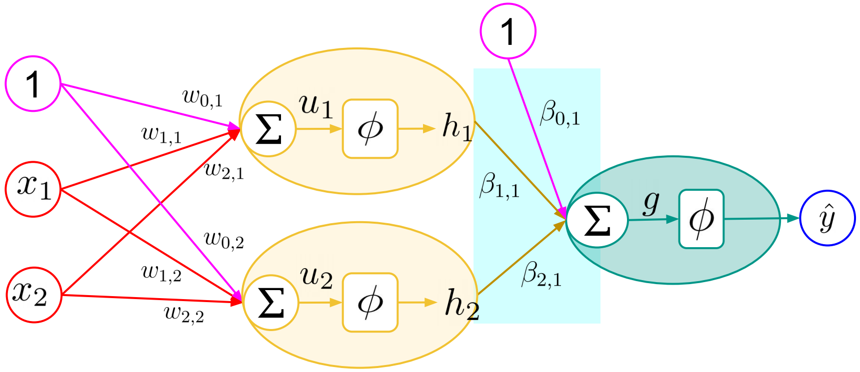
Weight adjustment

Output weight adjustment

$$\beta = \beta - \eta \nabla_{\beta} E$$

Input/Hidden weight adjustment

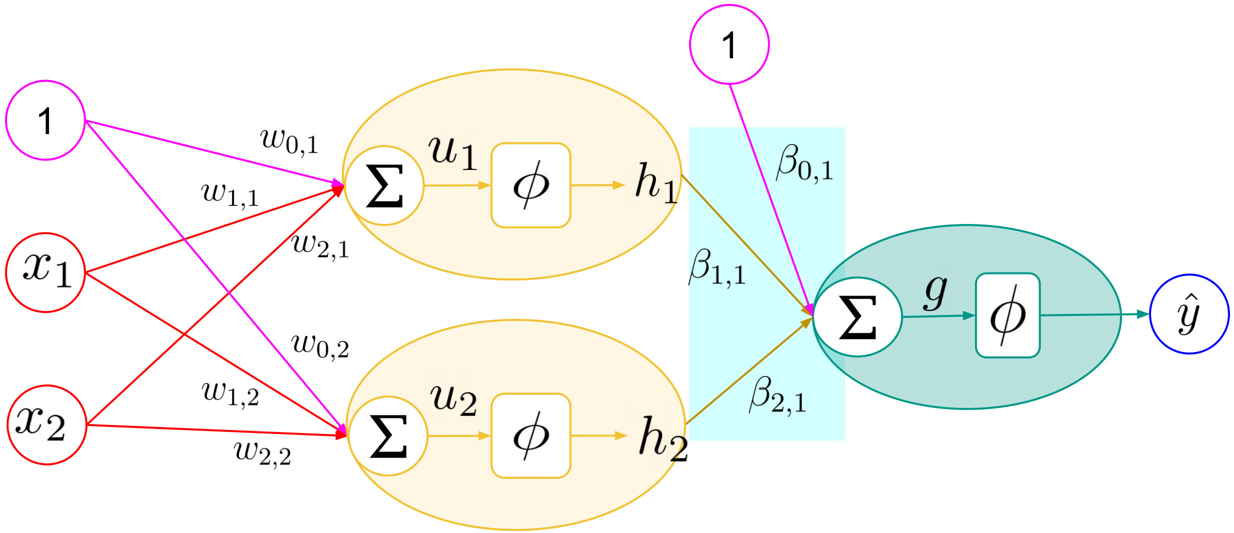
$$w = w - \eta \nabla_{w} E$$



Output weight adjustment

$$\beta = \beta - \eta \nabla_{\beta} E$$

Output weights adjustment



Output weights adjustment

$$\beta = \beta - \eta \nabla_{\beta} E$$

$$\nabla_{\beta} E = \frac{\partial E}{\partial \beta}$$

$$\frac{\partial E}{\partial \beta} = \frac{1}{2} \frac{\partial e^2}{\partial e} \frac{\partial e}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial g} \frac{\partial g}{\partial \beta}$$

Recall:

$$E(t) = e^2(t)$$

$$e = y - \hat{y}$$

$$\hat{y} = \phi(g)$$

$$g = \beta_{0,1} + \sum_{k=1}^{l} \beta_{k,1} h_k$$

$$u_1 = w_{0,1} + \sum_{j=1}^{n} w_{j,1} x_j$$

$$u_2 = w_{0,2} + \sum_{j=1}^{n} w_{j,2} x_j$$

$$h_1 = \phi(u_1)$$

$$h_2 = \phi(u_2)$$

Output weights adjustment

$$\frac{\partial E}{\partial \beta} = \frac{1}{2}\frac{\partial e^2}{\partial e}\frac{\partial e}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial g}\frac{\partial g}{\partial \beta}$$

Recall:

$$E(t) = e^2(t)$$
$$e = y - \hat{y}$$

$$\hat{y} = \phi(g)$$

$$g = \beta_{0,1} + \sum_{k=1}^{l} \beta_{k,1} h_k$$

$$u_1 = w_{0,1} + \sum_{j=1}^{n} w_{j,1} x_j$$

$$u_2 = w_{0,2} + \sum_{j=1}^{n} w_{j,2} x_j$$

$$h_1 = \phi(u_1)$$

$$h_2 = \phi(u_2)$$

Output weights adjustment

$$\frac{\partial E}{\partial \beta} = \frac{1}{2}\frac{\partial e^2}{\partial e}\frac{\partial e}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial g}\frac{\partial g}{\partial \beta}$$

$$= -e\frac{\partial \hat{y}}{\partial g}h$$

$$= -e(\hat{y}(1-\hat{y}))h$$

$$\beta = \beta + \eta \boxed{e(\hat{y}(1-\hat{y}))h}$$

Recall:

$$E(t) = e^2(t)$$

$$e = y - \hat{y}$$

$$\hat{y} = \phi(g)$$

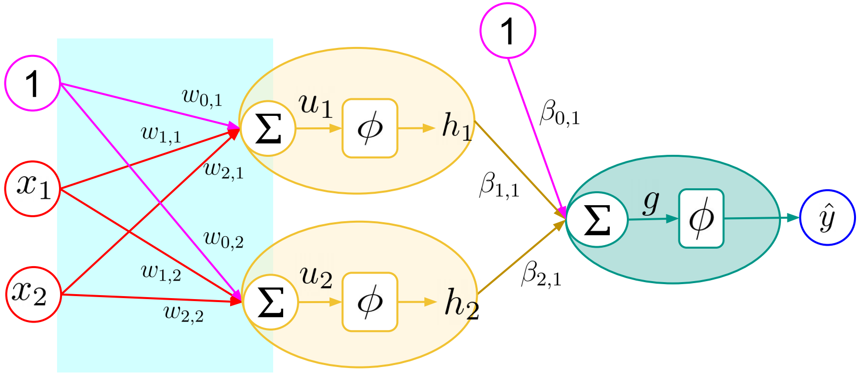
$$g = \beta_{0,1} + \sum_{k=1}^{l} \beta_{k,1} h_k$$

$$u_1 = w_{0,1} + \sum_{j=1}^{n} w_{j,1} x_j$$

$$u_2 = w_{0,2} + \sum_{j=1}^{n} w_{j,2} x_j$$

$$h_1 = \phi(u_1)$$

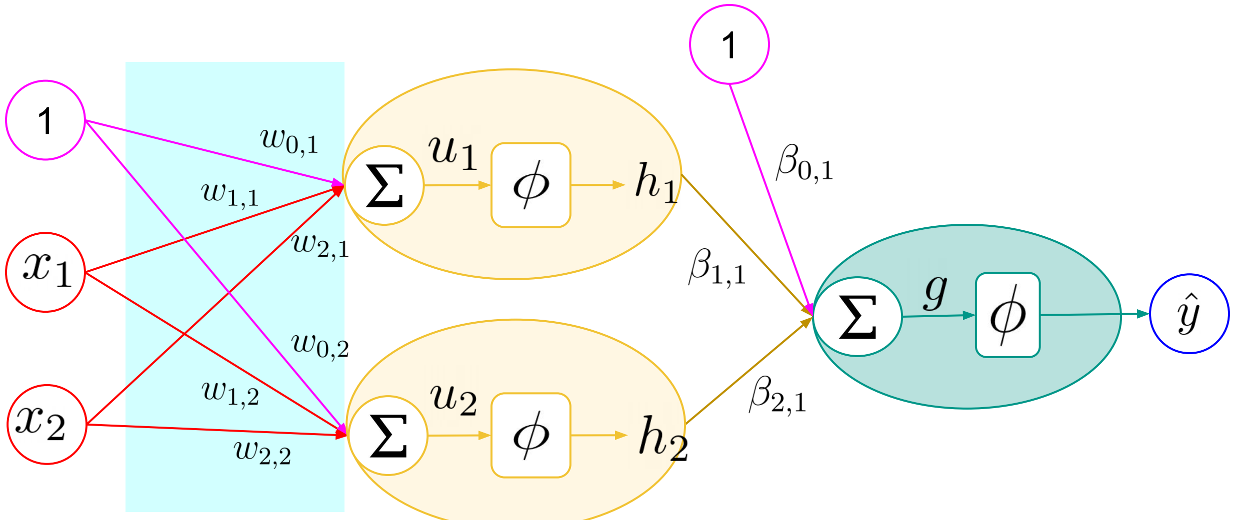
$$h_2 = \phi(u_2)$$



Input/Hidden weight adjustment

$$w = w - \eta \nabla_w E$$

Input/Hidden weights adjustment



Input/Hidden weights adjustment

$$w = w - \eta \nabla_w E$$

$$\nabla_w E = \frac{\partial E}{\partial w}$$

$$\frac{\partial E}{\partial w} = \frac{1}{2} \frac{\partial e^2}{\partial e} \frac{\partial e}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial g} \frac{\partial g}{\partial h} \frac{\partial h}{\partial u} \frac{\partial u}{\partial w}$$

Recall:

$$E(t) = e^2(t)$$

$$e = y - \hat{y}$$

$$\hat{y} = \phi(g)$$

$$g = \beta_{0,1} + \sum_{k=1}^{l} \beta_{k,1} h_k$$

$$u_1 = w_{0,1} + \sum_{j=1}^{n} w_{j,1} x_j$$

$$u_2 = w_{0,2} + \sum_{j=1}^{n} w_{j,2} x_j$$

$$h_1 = \phi(u_1)$$

$$h_2 = \phi(u_2)$$

Input/Hidden weights adjustment

$$\frac{\partial E}{\partial w} = \frac{1}{2}\frac{\partial e^2}{\partial e}\frac{\partial e}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial g}\frac{\partial g}{\partial h}\frac{\partial h}{\partial u}\frac{\partial u}{\partial w}$$

Recall:

$$E(t) = e^2(t)$$
$$e = y - \hat{y}$$

$$\hat{y} = \phi(g)$$

$$g = \beta_{0,1} + \sum_{k=1}^{l} \beta_{k,1} h_k$$

$$u_1 = w_{0,1} + \sum_{j=1}^{n} w_{j,1} x_j$$
$$u_2 = w_{0,2} + \sum_{j=1}^{n} w_{j,2} x_j$$
$$h_1 = \phi(u_1)$$
$$h_2 = \phi(u_2)$$

Input/Hidden weights adjustment

$$\frac{\partial E}{\partial w} = \frac{1}{2}\frac{\partial e^2}{\partial e}\frac{\partial e}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial g}\frac{\partial g}{\partial h}\frac{\partial h}{\partial u}\frac{\partial u}{\partial w}$$

$$= -e\frac{\partial \hat{y}}{\partial g}\beta\frac{\partial h}{\partial u}x$$

$$= -e(\hat{y}(1-\hat{y}))\beta(h(1-h))x$$

$$w = w + \eta \boxed{e(\hat{y}(1-\hat{y}))\beta(h(1-h))x}$$

Recall:

$$E(t) = e^2(t)$$

$$e = y - \hat{y}$$

$$\hat{y} = \phi(g)$$

$$g = \beta_{0,1} + \sum_{k=1}^{l} \beta_{k,1} h_k$$

$$u_1 = w_{0,1} + \sum_{j=1}^{n} w_{j,1} x_j$$

$$u_2 = w_{0,2} + \sum_{j=1}^{n} w_{j,2} x_j$$

$$h_1 = \phi(u_1)$$

$$h_2 = \phi(u_2)$$

Example: Train MLP for XOR

XOR gate

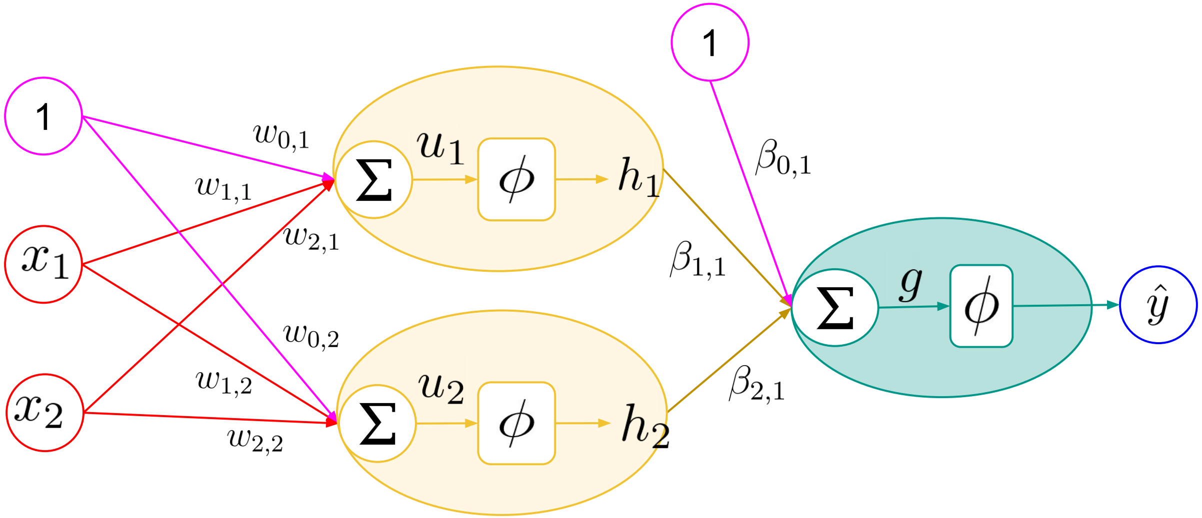
x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

Train MLP for XOR

Step 1:

- Define network architecture: 2-2-1
- Set the initial weights
- Define Activation function : "sigmoid" $\frac{1}{1+e^{-u(t)}}$
- Define the value of learning rate : 0.0001
- Define the stopping criteria i.e. *number of round* : 3

Train MLP for XOR



$$W = \begin{bmatrix} w_{0,1} = 1 & w_{0,2} = 2 \\ w_{1,1} = 1 & w_{1,2} = 2 \\ w_{2,1} = -2 & w_{2,2} = -1 \end{bmatrix}$$

$$\beta = \begin{bmatrix} \beta_{0,1} = 2 & \beta_{1,1} = -1 & \beta_{2,1} = 2 \end{bmatrix}$$

Train MLP for XOR

Step 2: Train model

- For each data point $(\mathbf{x})$

Feed-forward

- **Step 2.1:** Compute outputs of hidden layer
- **Step 2.2:** Compute outputs of output layer

$$u_1 = w_{0,1} + \sum_{j=1}^{n} w_{j,1} x_j$$

$$u_2 = w_{0,2} + \sum_{j=1}^{n} w_{j,2} x_j$$

$$h_1 = \phi(u_1)$$

$$h_2 = \phi(u_2)$$

Activation function: Sigmoid

$$\frac{1}{1+e^{-u(t)}}$$

$$g = \beta_{0,1} + \sum_{k=1}^{l} \beta_{k,1} h_k$$

$$\hat{y} = \phi(g)$$

Backpropagation

- **Step 2.3:** Adjust the weights of output layer
- **Step 2.4:** Adjust the weights of input (hidden) layer

$$\beta = \beta - \eta \nabla_{\beta} E$$

$$w = w - \eta \nabla_{w} E$$

Compute Feed-Forward

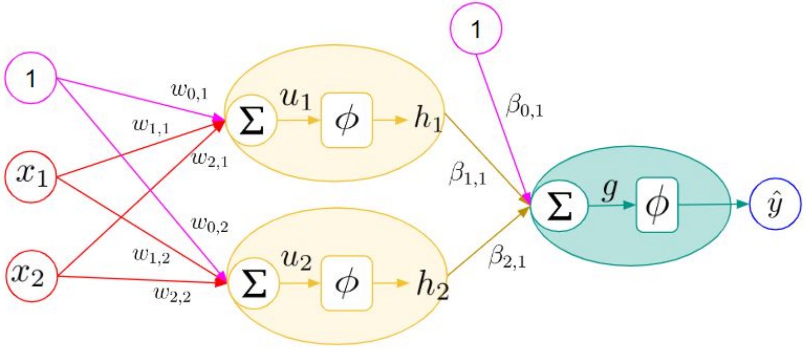
Feed-Forward

Step 2: Train model

- For each data point ($\mathbf{x}$)

Feed-forward

- **Step 2.1:** Compute outputs of hidden layer
- **Step 2.2:** Compute outputs of output layer



$$u_1 = w_{0,1} + \sum_{j=1}^{n} w_{j,1} x_j$$

$$u_2 = w_{0,2} + \sum_{j=1}^{n} w_{j,2} x_j$$

$$h_1 = \phi(u_1)$$

$$h_2 = \phi(u_2)$$

$$g = \beta_{0,1} + \sum_{k=1}^{l} \beta_{k,1} h_k$$

Activation function: Sigmoid

$$\frac{1}{1 + e^{-u(t)}}$$

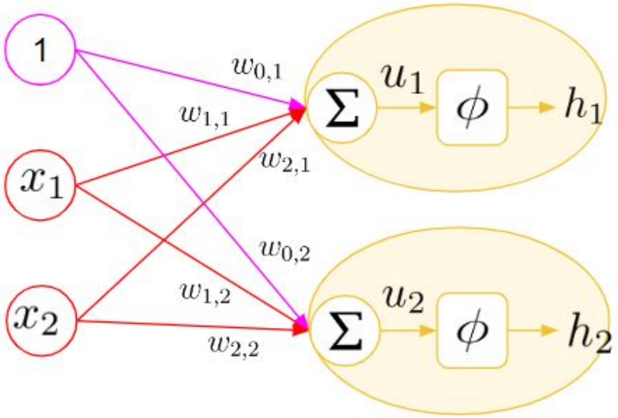
$$\hat{y} = \phi(g)$$

Feed-Forward

Step 2: Train Model

Round: 1
Learn with data row: 1

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



$$W = \begin{bmatrix} w_{0,1} = 1 & w_{0,2} = 2 \\ w_{1,1} = 1 & w_{1,2} = 2 \\ w_{2,1} = -2 & w_{2,2} = -1 \end{bmatrix}$$

Step 2.1: Compute outputs of hidden layer

$$u_1 = w_{0,1} + w_{1,1}x_1 + w_{2,1}x_2$$
$$u_2 = w_{0,2} + w_{1,2}x_1 + w_{2,2}x_2$$

$$u_1 = 1 + (1 \times 0) + (-2 \times 0) = 1$$
$$u_2 = 2 + (2 \times 0) + (-1 \times 0) = 2$$

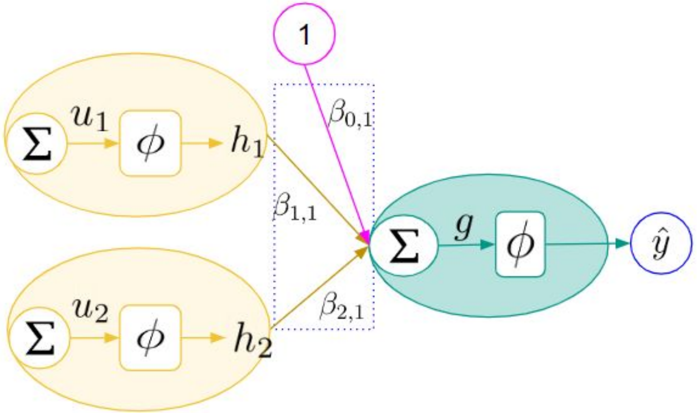
$$h_1 = \frac{1}{1 + e^{-u_1}} = \frac{1}{1 + e^{-1}} = 0.73$$
$$h_2 = \frac{1}{1 + e^{-u_2}} = \frac{1}{1 + e^{-2}} = 0.88$$

Feed-Forward

Step 2: Train Model

Round: 1
Learn with data row: 1

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



$$\beta = [\beta_{0,1} = 2 \quad \beta_{1,1} = -1 \quad \beta_{2,1} = 2]$$

Step 2.2: Compute outputs of output layer

$$g = \beta_{0,1} + \beta_{1,1} h_1 + \beta_{2,1} h_2$$

$$g = 2 + (-1 \times 0.73) + (2 \times 0.88)$$

$$= 3.03$$

$$\hat{y} = \frac{1}{1 + e^{-g}} = \frac{1}{1 + e^{-3.03}} = 0.95$$

Feed-Forward

Step 2: Train Model

Round: 1
Learn with data row: 1

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

Step 2.1: Compute outputs of hidden layer

$$u_1 = w_{0,1} + w_{1,1}x_1 + w_{2,1}x_2$$
$$u_2 = w_{0,2} + w_{1,2}x_1 + w_{2,2}x_2$$

$$u_1 = 1 + (1 \times 0) + (-2 \times 0) = 1$$
$$u_2 = 2 + (2 \times 0) + (-1 \times 0) = 2$$

$$h_1 = \frac{1}{1+e^{-u_1}} = \frac{1}{1+e^{-1}} = 0.73$$
$$h_2 = \frac{1}{1+e^{-u_2}} = \frac{1}{1+e^{-2}} = 0.88$$

Step 2.2: Compute outputs of output layer

$$g = \beta_{0,1} + \beta_{1,1}h_1 + \beta_{2,1}h_2$$

$$g = 2 + (-1 \times 0.73) + (2 \times 0.88)$$
$$= 3.03$$

$$\hat{y} = \frac{1}{1+e^{-g}} = \frac{1}{1+e^{-3.03}} = 0.95$$

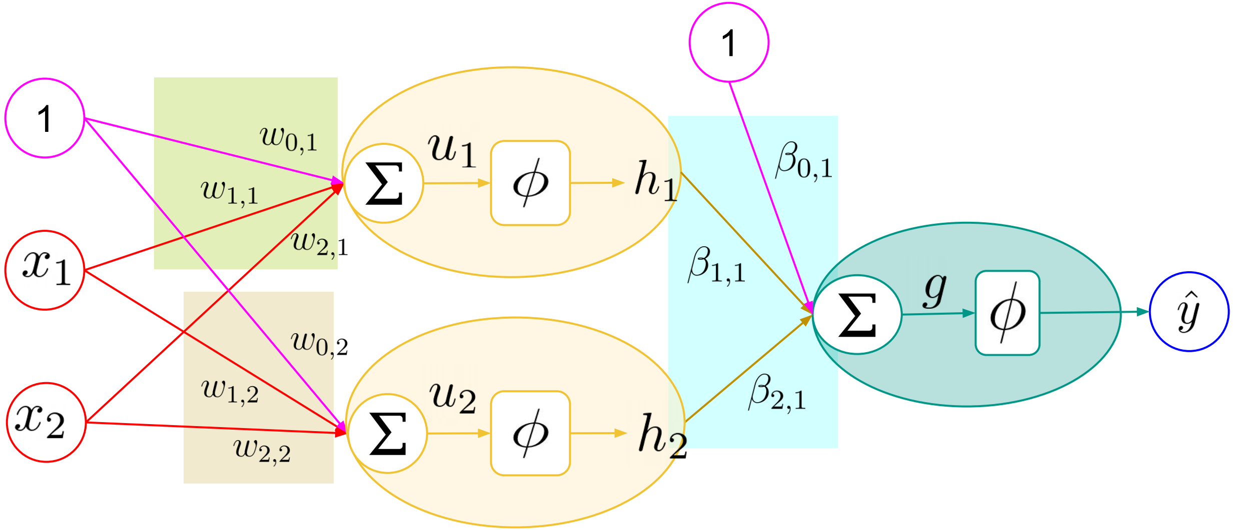
Update Weights by Backpropagation

Backpropagation

Step 2: Train Model

Round: 1
Learn with data row: 1

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



Backpropagation

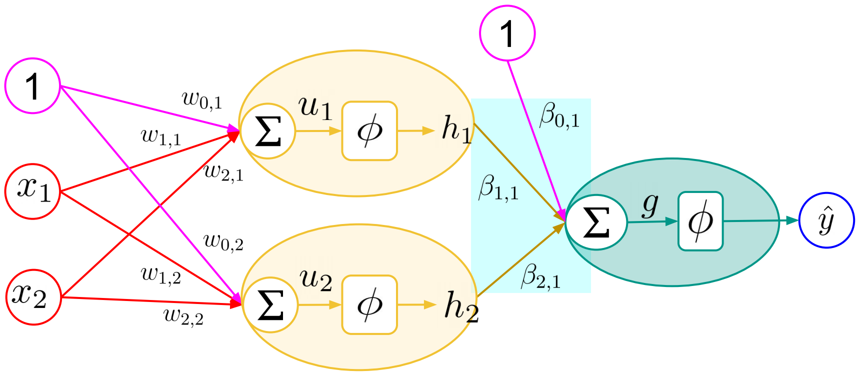
Step 2: Train Model

Round: 1
Learn with data row: 1

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

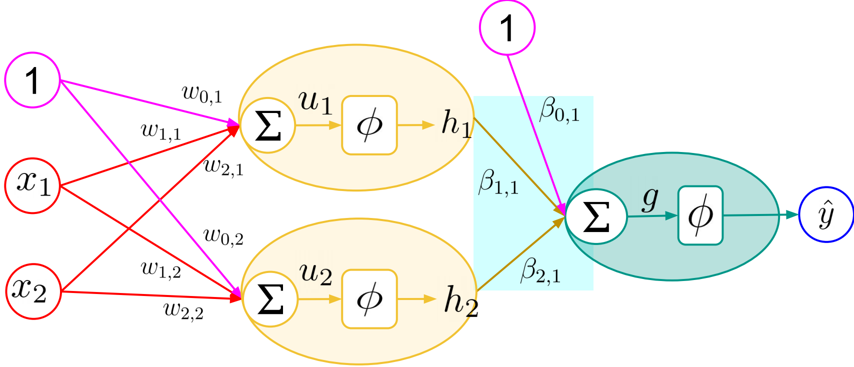
Output weights tuning

$$\beta = \beta + \eta e(\hat{y}(1-\hat{y}))h$$



Backpropagation

Step 2: Train Model



Round: 1
Learn with data row: 1

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

Output weights tuning

$$\beta = \beta + \eta e(\hat{y}(1-\hat{y}))h$$

$$\beta_{0,1} = \beta_{0,1} + (\eta)(y-\hat{y})(\hat{y}(1-\hat{y}))(1)$$
$$\beta_{0,1} = 2 + (0.0001)(0-0.95)(0.95(1-0.95))(1) \simeq 1.99$$

$$\beta_{1,1} = \beta_{1,1} + (\eta)(y-\hat{y})(\hat{y}(1-\hat{y}))(h_1)$$
$$\beta_{1,1} = -1 + (0.0001)(0-0.95)(0.95(1-0.95))(0.73) \simeq -1$$

$$\beta_{2,1} = \beta_{2,1} + (\eta)(y-\hat{y})(\hat{y}(1-\hat{y}))(h_2)$$
$$\beta_{2,1} = 2 + (0.0001)(0-0.95)(0.95(1-0.95))(0.88) \simeq 1.99$$

Backpropagation

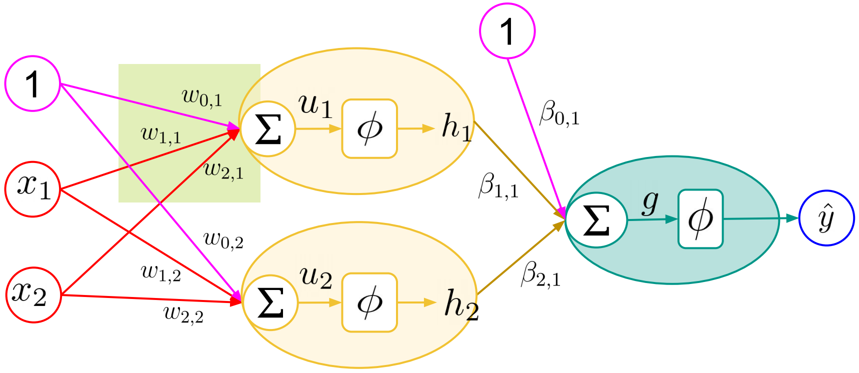
Step 2: Train Model

Round: 1
Learn with data row: 1

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

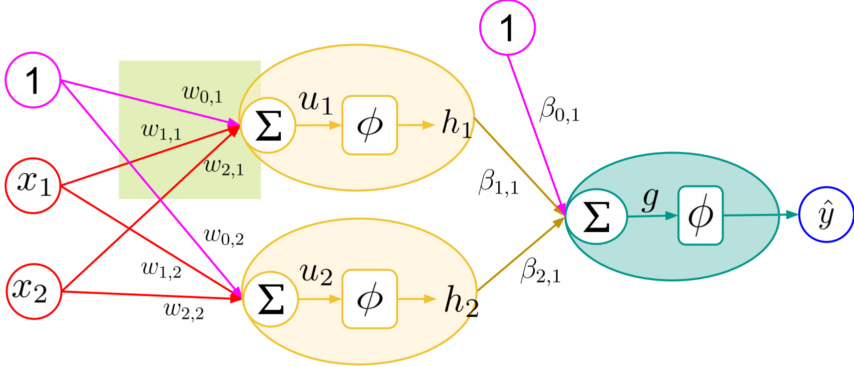
Input weights tuning

$$w = w + \eta e(\hat{y}(1-\hat{y}))\beta(h(1-h))x$$



Backpropagation

Step 2: Train Model



Round: 1
Learn with data row: 1

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

Input weights tuning

$$\boxed{w = w + \eta e(\hat{y}(1-\hat{y}))\beta(h(1-h))x}$$

$$w_{0,1} = w_{0,1} + \eta(y-\hat{y})(\hat{y}(1-\hat{y}))\beta_{1,1}(h_1(1-h_1))(1)$$
$$w_{0,1} = 1 + (0.0001)(0-0.95)(0.95(1-0.95))(-1)(0.73(1-0.73))(1) \simeq 1$$

$$w_{1,1} = w_{1,1} + \eta(y-\hat{y})(\hat{y}(1-\hat{y}))\beta_{1,1}(h_1(1-h_1))x_1$$
$$w_{1,1} = 1 + (0.0001)(0-0.95)(0.95(1-0.95))(-1)(0.73(1-0.73))0 \simeq 1$$

$$w_{2,1} = w_{2,1} + \eta(y-\hat{y})(\hat{y}(1-\hat{y}))\beta_{1,1}(h_1(1-h_1))x_2$$
$$w_{2,1} = -2 + (0.0001)(0-0.95)(0.95(1-0.95))(-1)(0.73(1-0.73))0 \simeq -2$$

Backpropagation

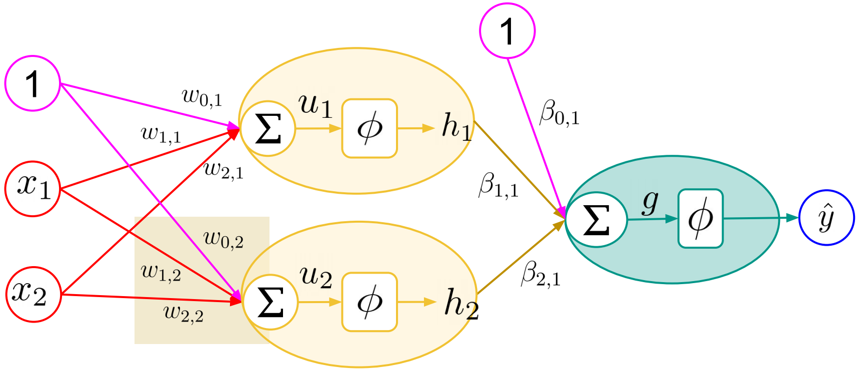
Step 2: Train Model

Round: 1
Learn with data row: 1

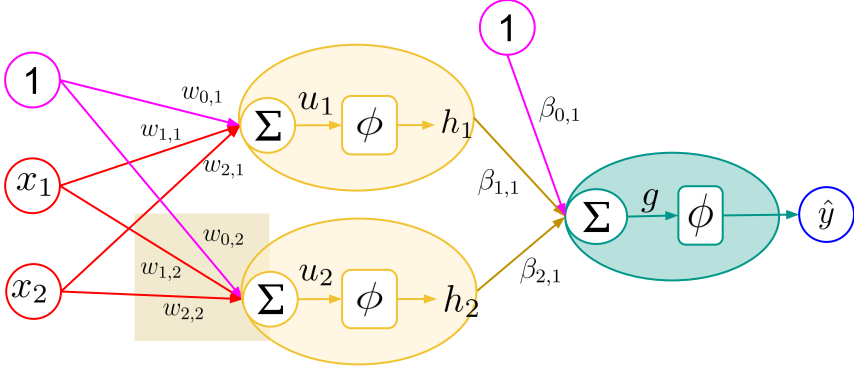
x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

Input weights tuning

$$w = w + \eta e(\hat{y}(1-\hat{y}))\beta(h(1-h))x$$



Backpropagation



Step 2: Train Model

Round: 1
Learn with data row: 1

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

Input weights tuning

$$\boxed{w = w + \eta e(\hat{y}(1-\hat{y}))\beta(h(1-h))x}$$

$$w_{0,2} = w_{0,2} + \eta(y-\hat{y})(\hat{y}(1-\hat{y}))\beta_{1,2}(h_2(1-h_2))(1)$$
$$w_{0,2} = 2 + (0.0001)(0-0.95)(0.95(1-0.95))(2)(0.88(1-0.88))(1) \simeq 1.99$$

$$w_{1,2} = w_{1,2} + \eta(y-\hat{y})(\hat{y}(1-\hat{y}))\beta_{1,2}(h_2(1-h_2))x_1$$
$$w_{1,2} = 2 + (0.0001)(0-0.95)(0.95(1-0.95))(2)(0.88(1-0.88))0 \simeq 2$$

$$w_{2,2} = w_{2,2} + \eta(y-\hat{y})(\hat{y}(1-\hat{y}))\beta_{1,2}(h_2(1-h_2))x_2$$
$$w_{2,2} = -1 + (0.0001)(0-0.95)(0.95(1-0.95))(2)(0.88(1-0.88))0 \simeq -1$$

Train MLP for XOR

Results After

Round: 1

Learn with data row: 1

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

$$W = \begin{bmatrix} w_{0,1} = 1 & w_{0,2} = 1.99 \\ w_{1,1} = 1 & w_{1,2} = 2 \\ w_{2,1} = -2 & w_{2,2} = -1 \end{bmatrix}$$

$$\beta = \begin{bmatrix} \beta_{1,0} = 1.99 & \beta_{1,1} = -1 & w_{1,2} = 1.99 \end{bmatrix}$$

Hands on

Train MLP for XOR

Hands On 1:

แสดงการคำนวณหา weight ที่ได้จากการ train model ด้วยข้อมูลแถวที่ 2,3,4

Round: **1**
Learn with data row: **2**

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

Round: **1**
Learn with data row: **3**

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

Round: **1**
Learn with data row: **4**

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

Train MLP for XOR

Hands On 2:

แสดงการคำนวณหา weight ที่ได้จากการ train model ด้วยข้อมูลแต่ละแถว ในรอบที่2 และค่า weight ชุดสุดท้ายที่คำนวณได้มีค่าเป็นเท่าไร

Round: **2**
Learn with data row: 1

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

Round: **2**
Learn with data row: 2

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

Round: **2**
Learn with data row: 3

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

Round: **2**
Learn with data row: 4

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