# Neural Network and Deep Learning

Random Neural Network

YANIKA KONGSOROT

# Outline

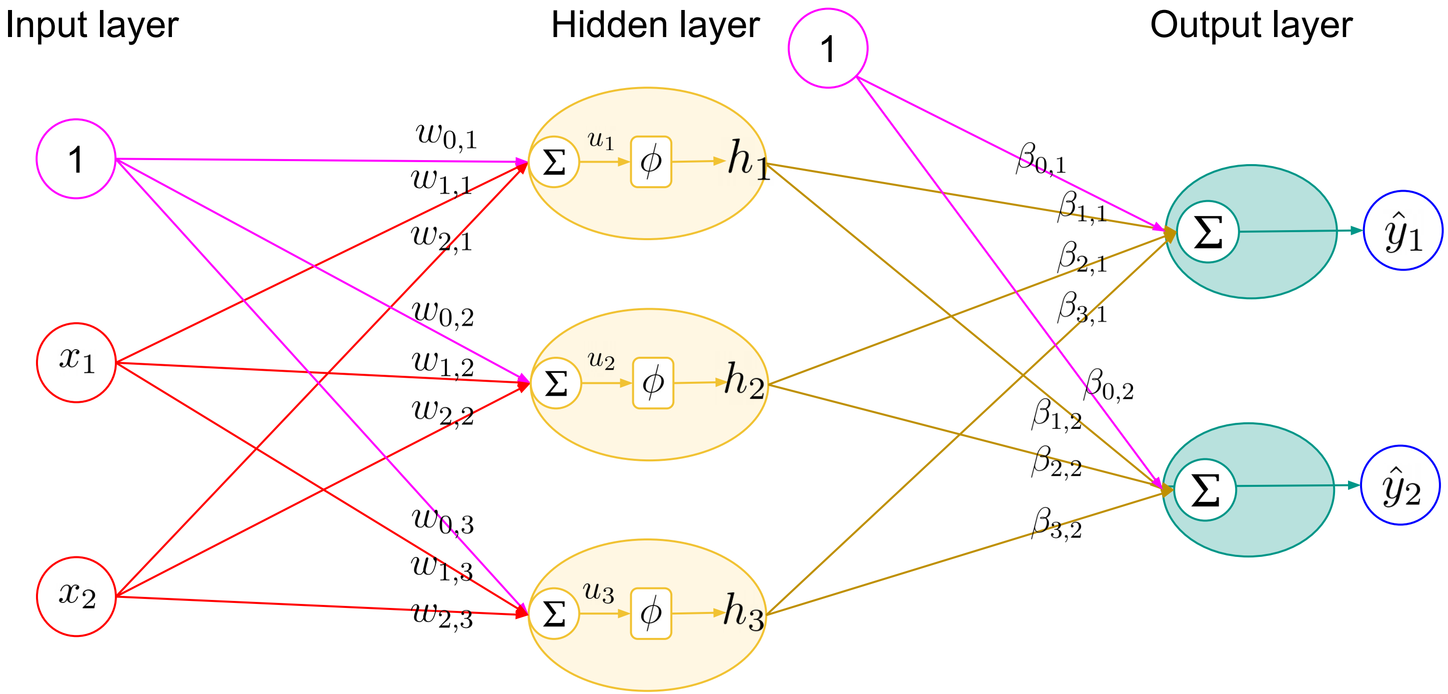
- Random Neural Network
- Learning algorithm of Random Neural Network

# Random Neural Network

# Random Neural Network

- Single hidden layer feedforward neural network
- **Input weights** are randomly chosen
- **Output weights** are analytically computed by the generalized Pseudo inverse matrix.
- No iterative tuning
- Fast learning model

# Single hidden layer feedforward neural network



# Parameters

# Input

**Data X**

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_N \end{bmatrix} = \begin{bmatrix} x_{1,1} & x_{1,2} \\ \vdots & \vdots \\ x_{N,1} & x_{N,2} \end{bmatrix}$$

**Input Data X:** Data + Bias

$$\mathbf{X} = \begin{bmatrix} 1 & \mathbf{x}_1 \\ \vdots & \vdots \\ 1 & \mathbf{x}_N \end{bmatrix} = \begin{bmatrix} 1 & x_{1,1} & x_{1,2} \\ \vdots & \vdots & \vdots \\ 1 & x_{N,1} & x_{N,2} \end{bmatrix}$$

# Target

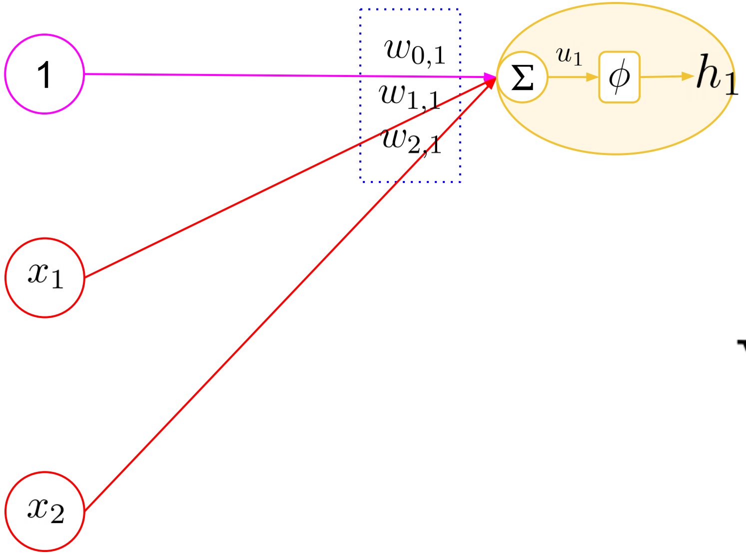
**Target Y**

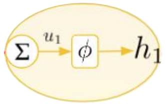
$$\mathbf{Y} = \begin{bmatrix} \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_N \end{bmatrix} = \begin{bmatrix} y_{1,1} & y_{1,2} \\ \vdots & \vdots \\ y_{N,1} & y_{N,2} \end{bmatrix}$$

**Predicted results of Y**

$$\hat{\mathbf{Y}} = \begin{bmatrix} \hat{\mathbf{y}}_1 \\ \vdots \\ \hat{\mathbf{y}}_N \end{bmatrix} = \begin{bmatrix} \hat{y}_{1,1} & \hat{y}_{1,2} \\ \vdots & \vdots \\ \hat{y}_{N,1} & \hat{y}_{N,2} \end{bmatrix}$$

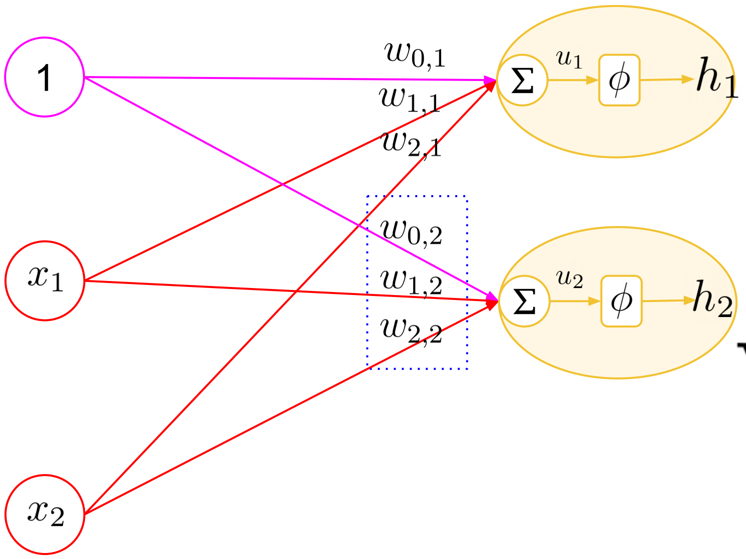
# Input Weights




$$\mathbf{w} = \begin{array}{c} 1 \\ x_1 \\ x_2 \end{array} \begin{bmatrix} w_{0,1} \\ w_{1,1} \\ w_{2,1} \end{bmatrix}$$

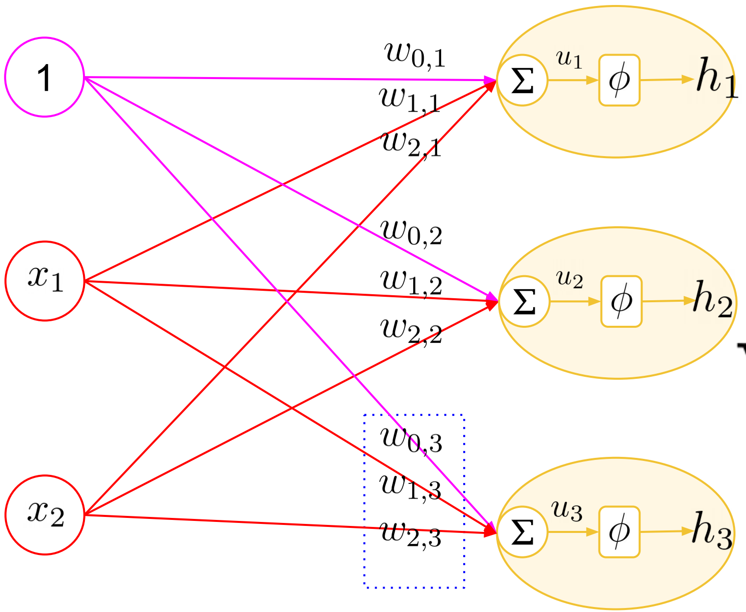
# Input Weights

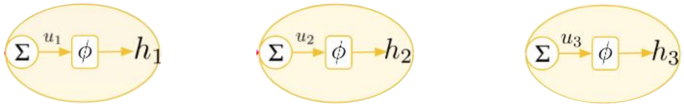




$$\mathbf{W} = \begin{array}{c} 1 \\ x_1 \\ x_2 \end{array} \begin{bmatrix} w_{0,1} & w_{0,2} \\ w_{1,1} & w_{1,2} \\ w_{2,1} & w_{2,2} \end{bmatrix}$$

# Input Weights



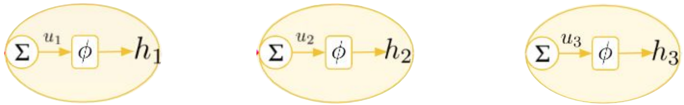




$$\mathbf{W} = \begin{array}{c} 1 \\ x_1 \\ x_2 \end{array} \begin{bmatrix} w_{0,1} & w_{0,2} & w_{0,3} \\ w_{1,1} & w_{1,2} & w_{1,3} \\ w_{2,1} & w_{2,2} & w_{2,3} \end{bmatrix}$$

# Input Weights

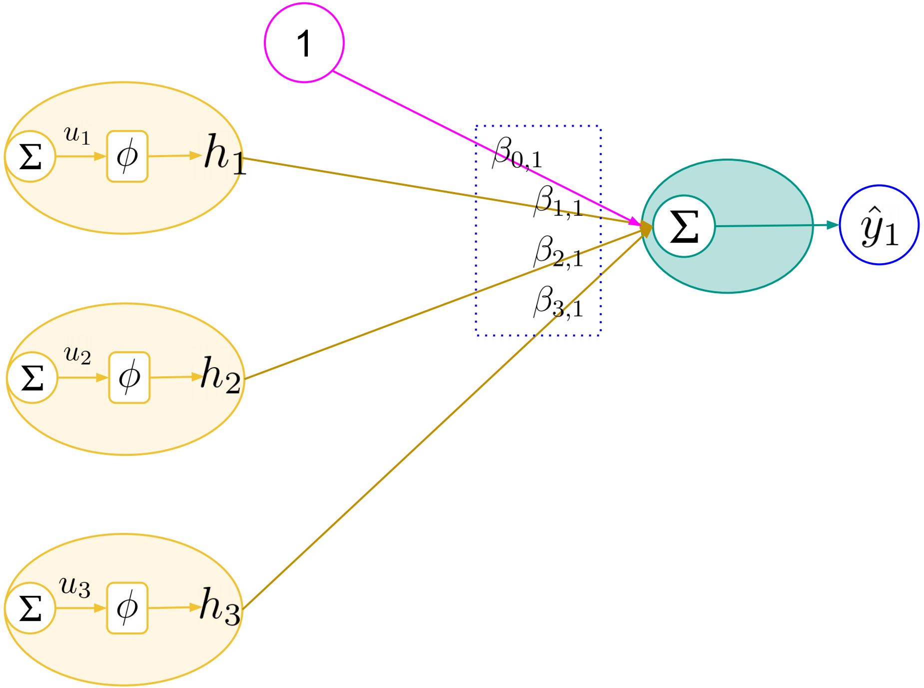
**Input Weights W**

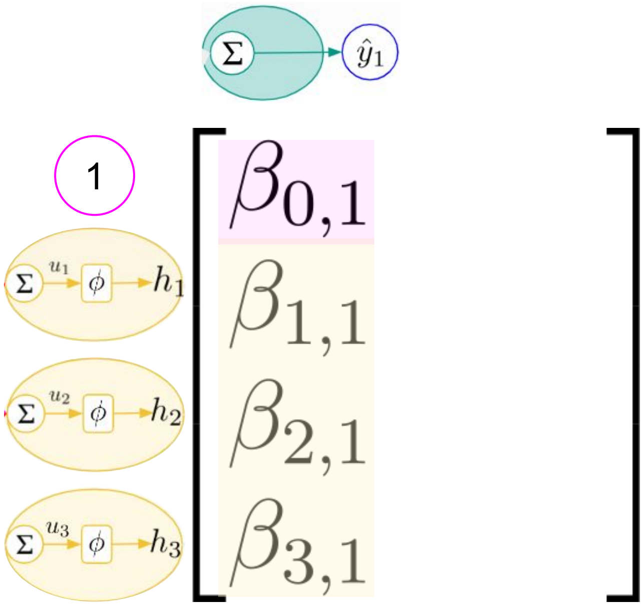
$$\mathbf{w} = \begin{bmatrix} w_{0,1} & w_{0,2} & w_{0,3} \\ w_{1,1} & w_{1,2} & w_{1,3} \\ w_{2,1} & w_{2,2} & w_{2,3} \end{bmatrix}$$



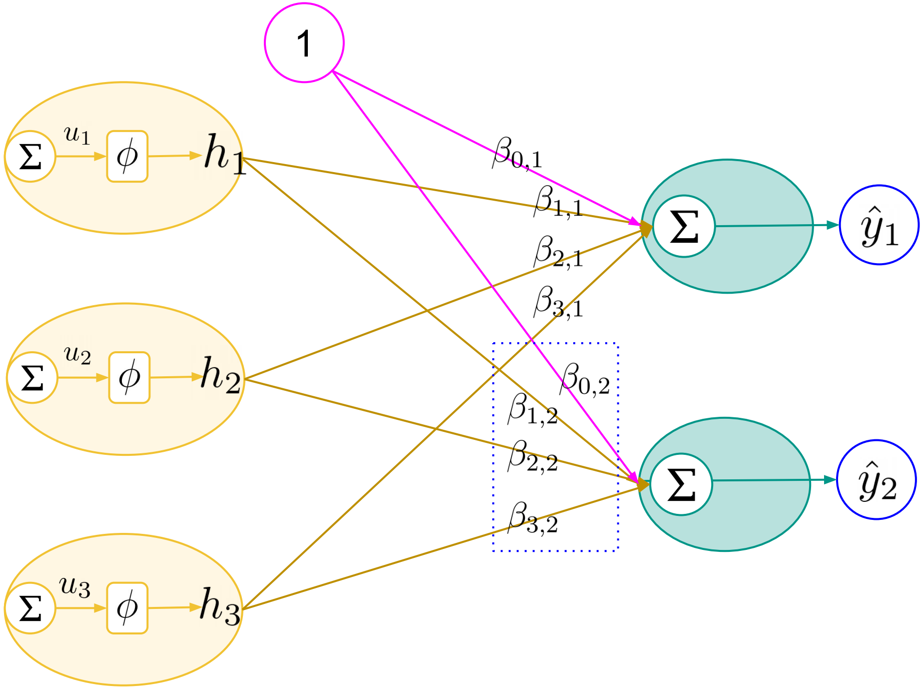

$$\begin{array}{c} 1 \\ x_1 \\ x_2 \end{array} \begin{bmatrix} w_{0,1} & w_{0,2} & w_{0,3} \\ w_{1,1} & w_{1,2} & w_{1,3} \\ w_{2,1} & w_{2,2} & w_{2,3} \end{bmatrix}$$

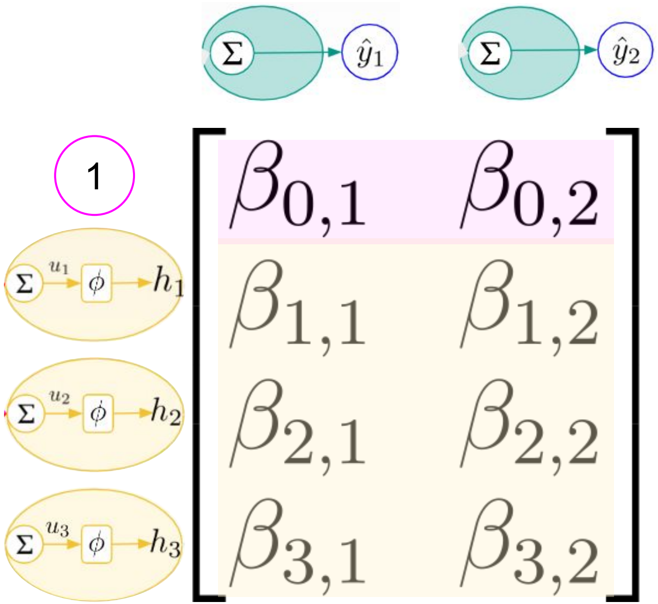
# Output Weights



# Output Weights

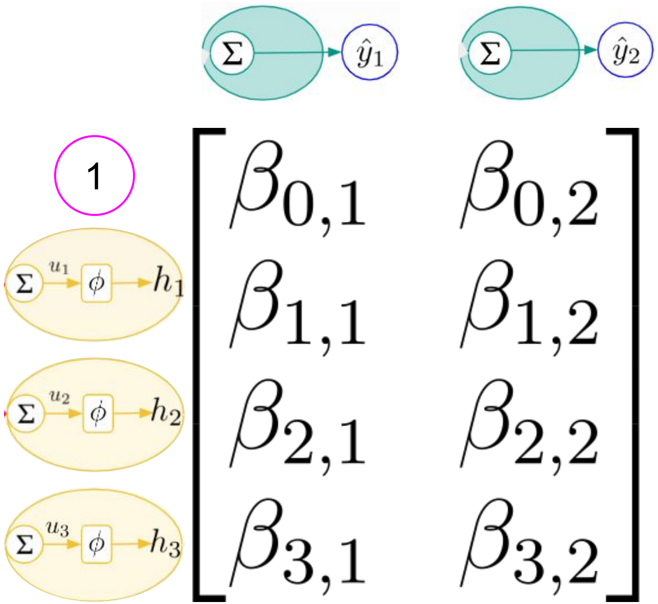


# Output Weights

**Output Weights**

$$\boldsymbol{\beta} = \begin{bmatrix} \beta_{0,1} & \beta_{0,2} \\ \beta_{1,1} & \beta_{1,2} \\ \beta_{2,1} & \beta_{2,2} \\ \beta_{3,1} & \beta_{3,2} \end{bmatrix}$$



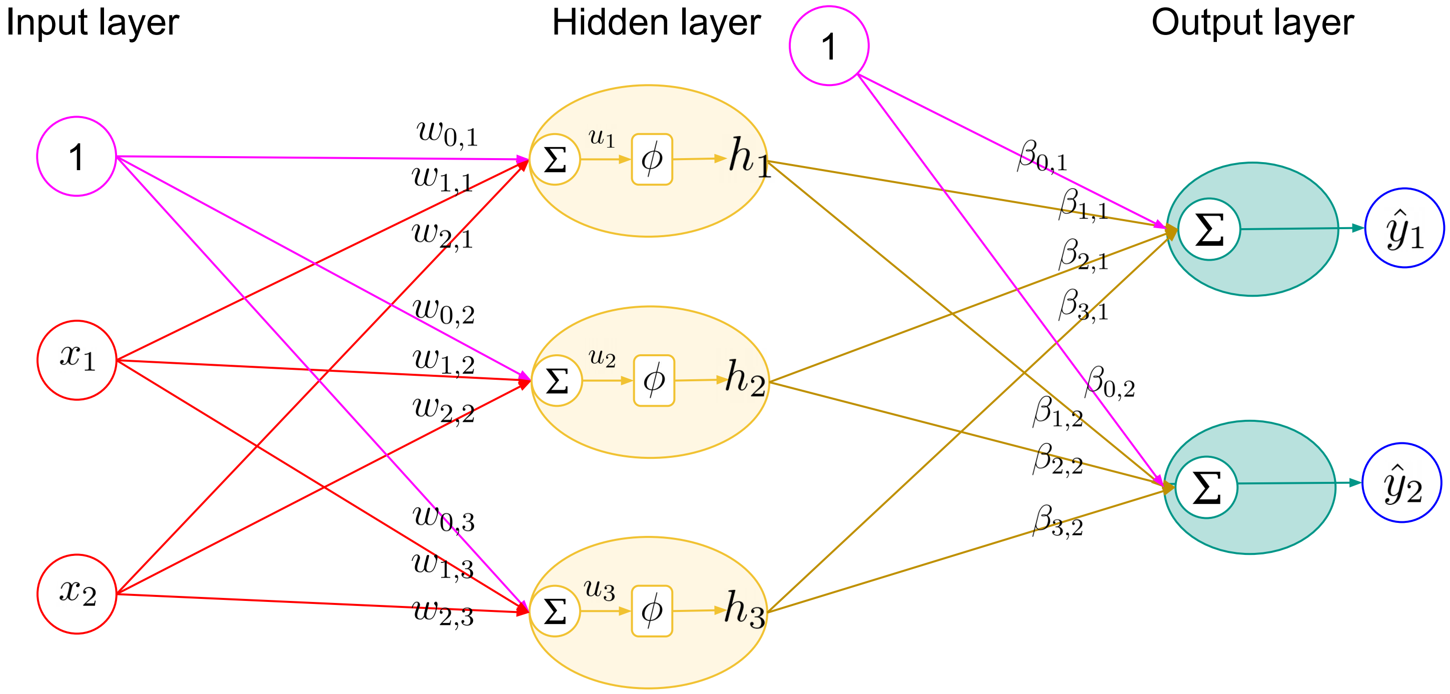
# Learning Algorithm

# Learning Algorithm

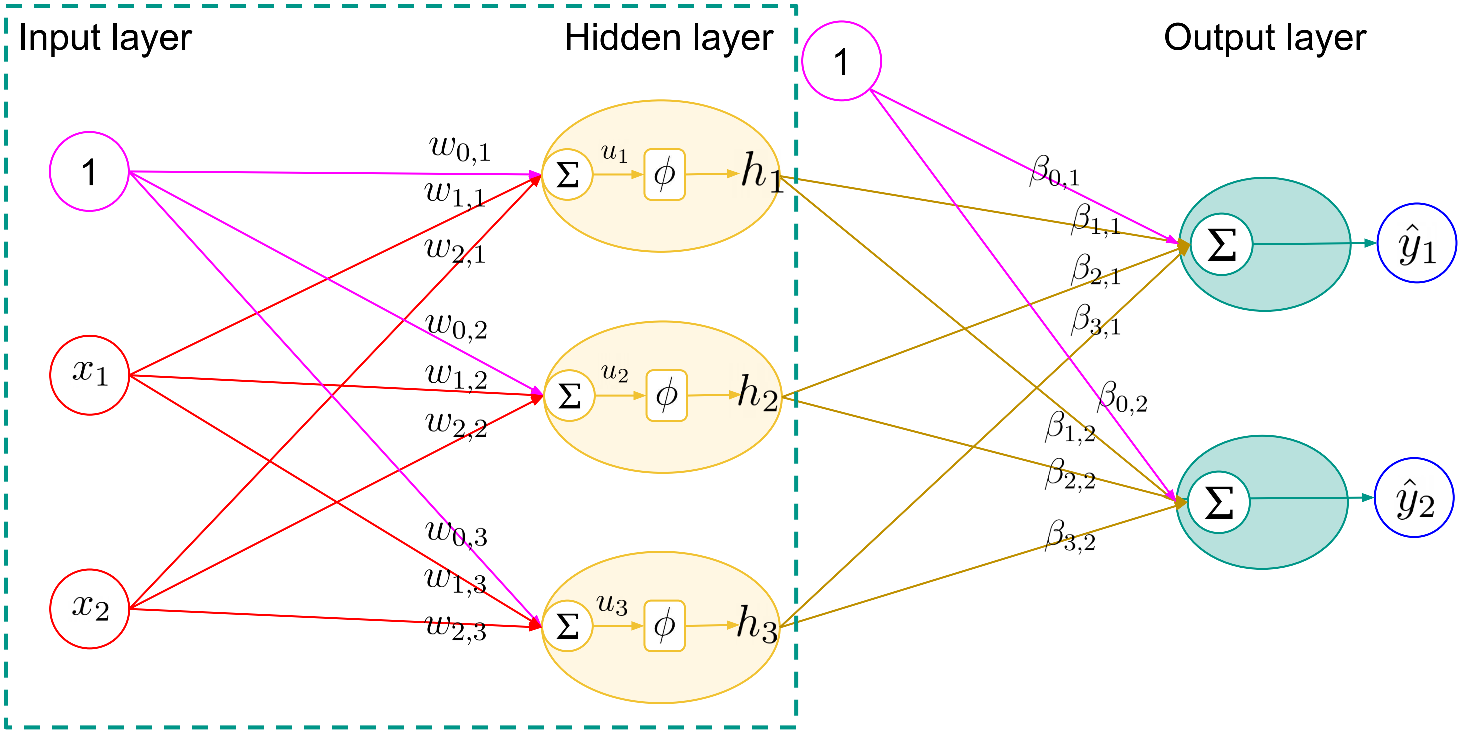
**Feed-forward Learning**

- Hidden layer calculation
  - Random input weights
  - Compute Hidden layer output
- Output layer calculation
  - Compute output weight using the **generalized Pseudo inverse**

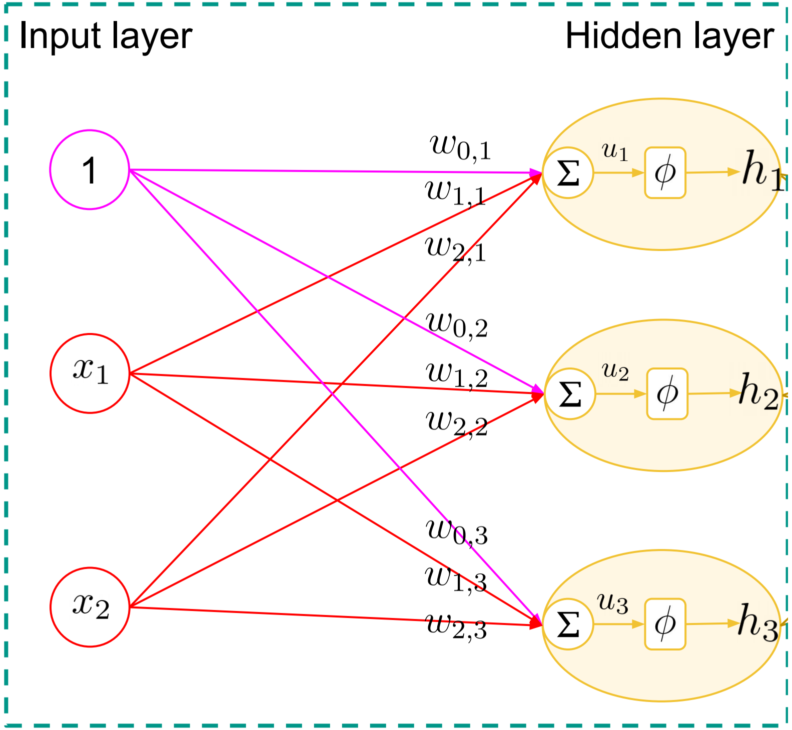
# Learning Algorithm



# Learning Algorithm

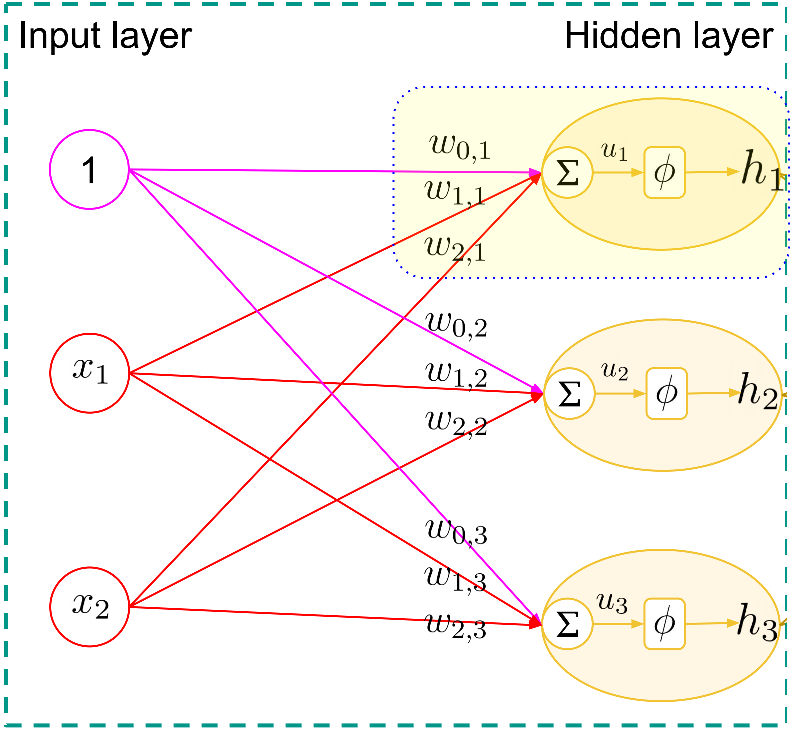


# Learning Algorithm




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

# Learning Algorithm




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

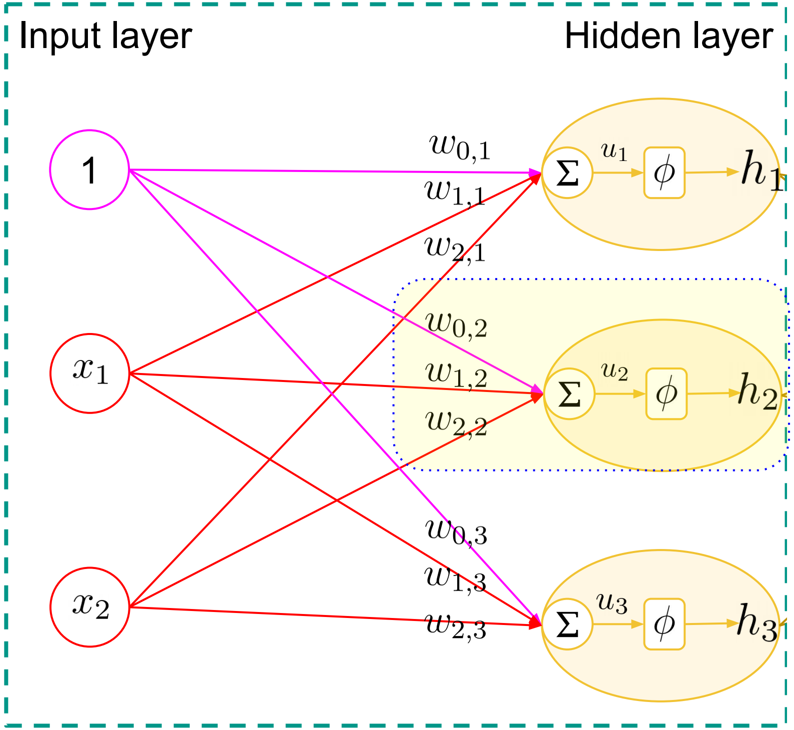
$$= \mathbf{x}_{(i,:)} \cdot \mathbf{w}_{(:,1)}$$

$$= \begin{bmatrix} 1 & x_{i,1} & x_{i,2} \end{bmatrix} \cdot \begin{bmatrix} w_{0,1} \\ w_{1,1} \\ w_{2,1} \end{bmatrix}$$

$$h_{(i,1)} = \phi\left(\mathbf{x}_{(i,:)} \cdot \mathbf{w}_{(:,1)}\right)$$

$i \in [1, \cdots, N]$

# Learning Algorithm

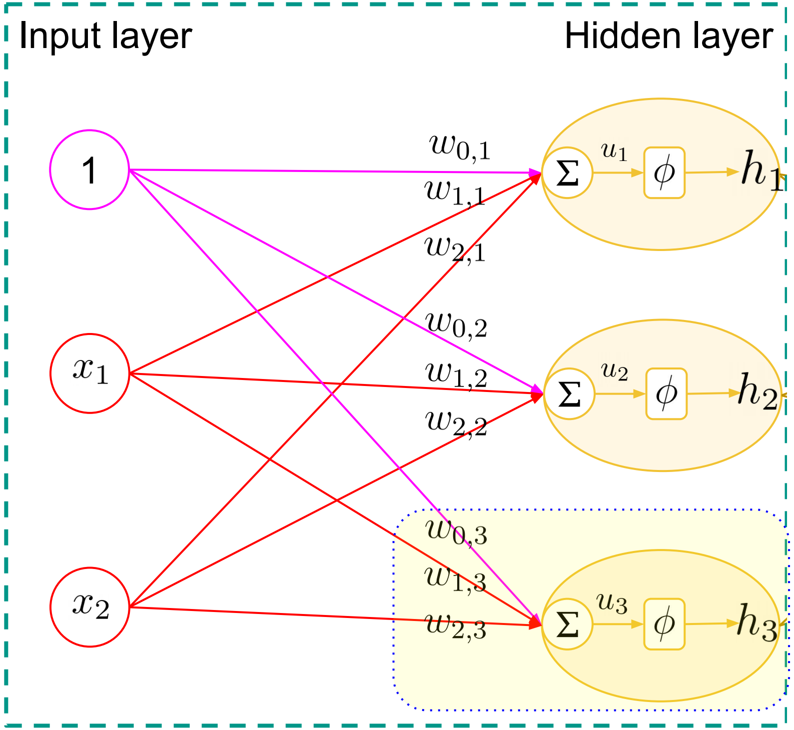



$$
\begin{aligned}
u_2 &= w_{0,2} + \sum_{j=1}^{n} w_{j,2} x_{i,j} \\
&= \mathbf{x}_{(i,:)} \cdot \mathbf{w}_{(:,2)} \\
&= \begin{bmatrix} 1 & x_{i,1} & x_{i,2} \end{bmatrix} \cdot \begin{bmatrix} w_{0,2} \\ w_{1,2} \\ w_{2,2} \end{bmatrix}
\end{aligned}
$$

$$h_{(i,2)} = \phi\left(\mathbf{x}_{(i,:)} \cdot \mathbf{w}_{(:,2)}\right)$$

$i \in [1, \cdots, N]$

# Learning Algorithm

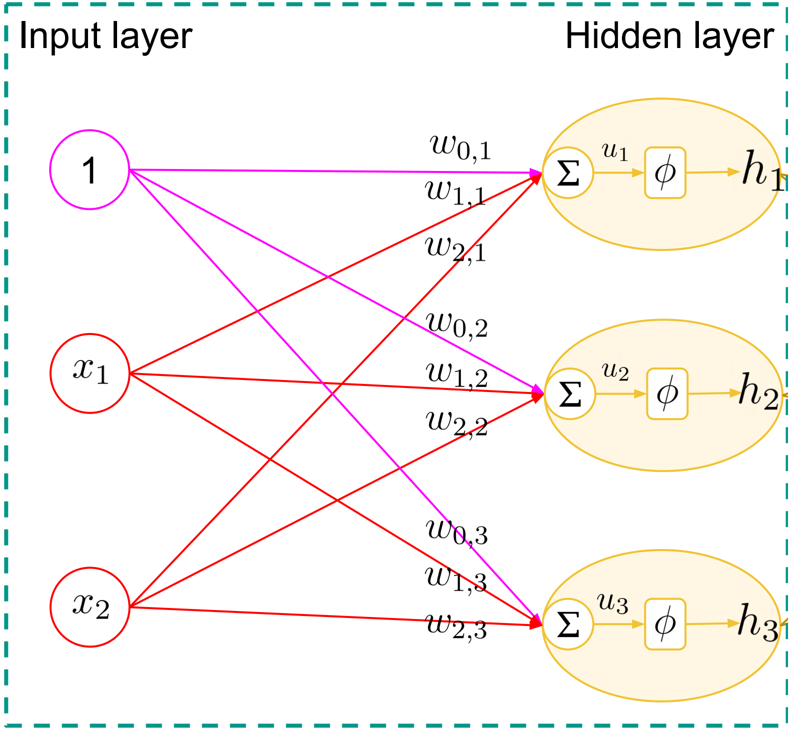



$$
\begin{aligned}
u_3 &= w_{0,3} + \sum_{j=1}^{n} w_{j,3} x_{i,j} \\
&= \mathbf{x}_{(i,:)} \cdot \mathbf{w}_{(:,3)} \\
&= \begin{bmatrix} 1 & x_{i,1} & x_{i,2} \end{bmatrix} \cdot \begin{bmatrix} w_{0,3} \\ w_{1,3} \\ w_{2,3} \end{bmatrix}
\end{aligned}
$$

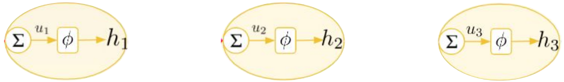
$$
h_{(i,3)} = \phi\left(\mathbf{x}_{(i,:)} \cdot \mathbf{w}_{(:,3)}\right)
$$

$i \in [1, \cdots, N]$

# Learning Algorithm



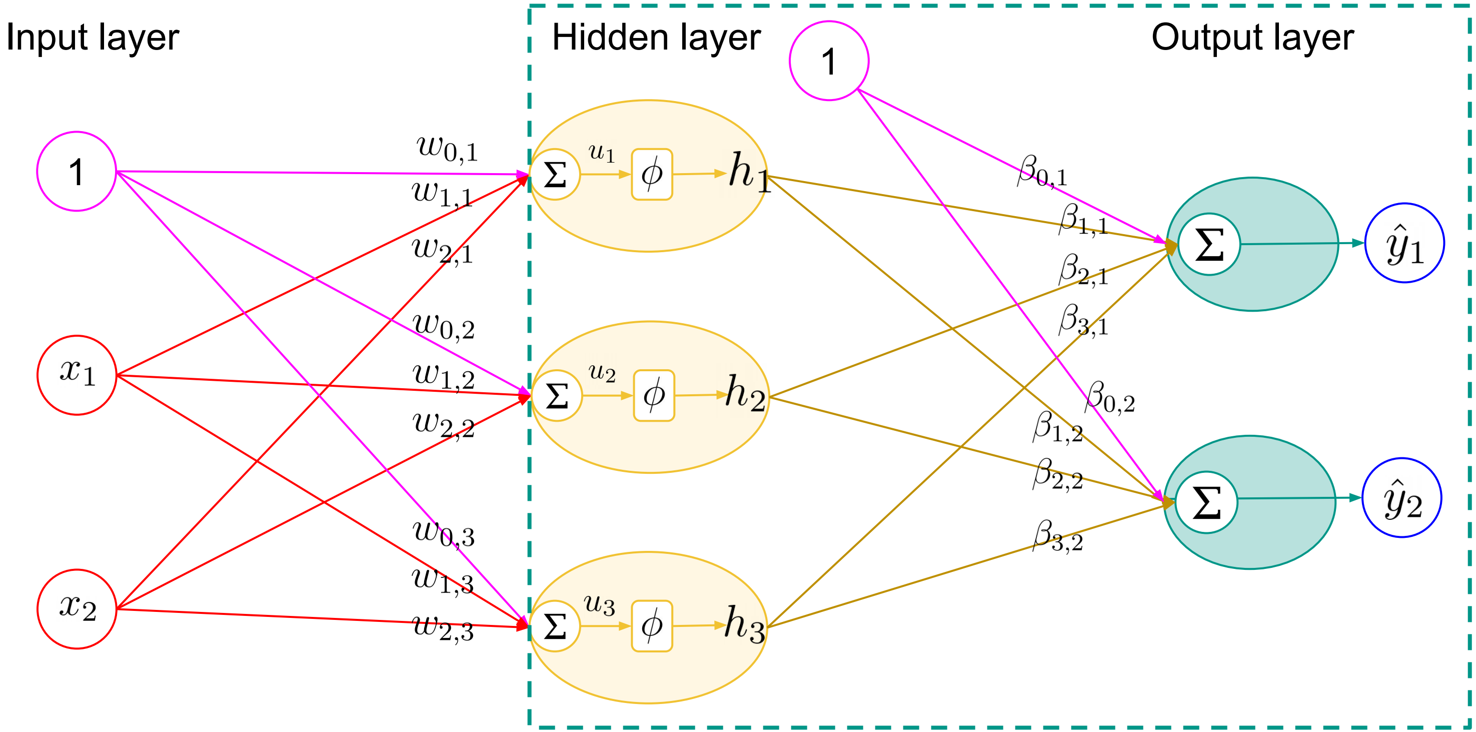

$$\mathbf{H} = \phi(\mathbf{X} \cdot \mathbf{W})$$



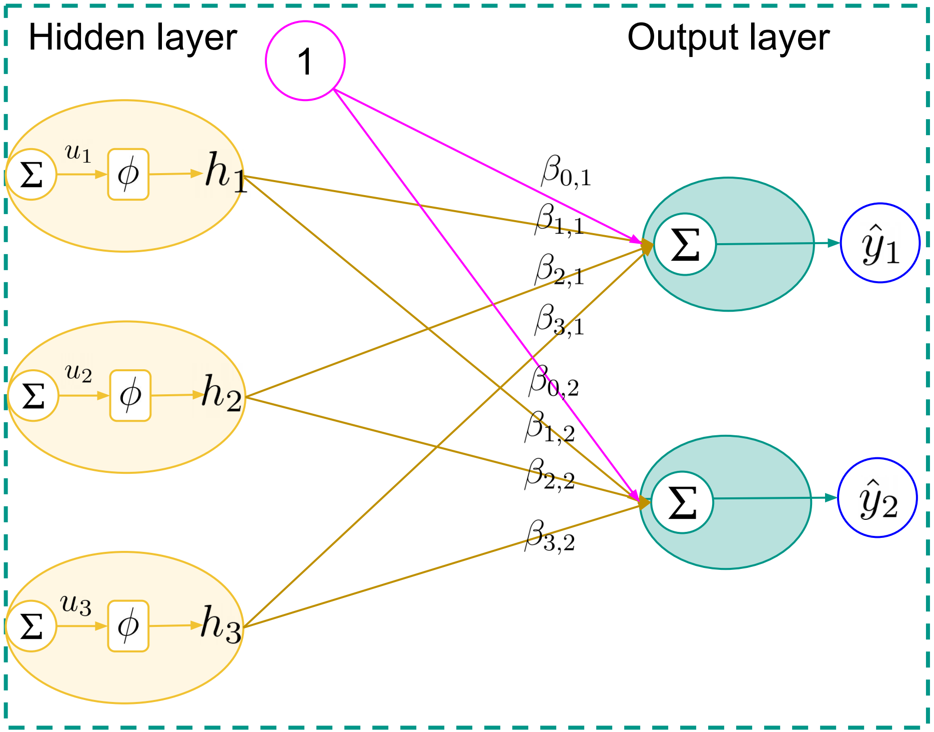

$$\mathbf{H} = \begin{bmatrix} h_{1,1} & h_{1,2} & h_{1,3} \\ \vdots & \vdots & \\ h_{N,1} & h_{N,2} & h_{N,3} \end{bmatrix}$$

$i \in [1, \cdots, N]$

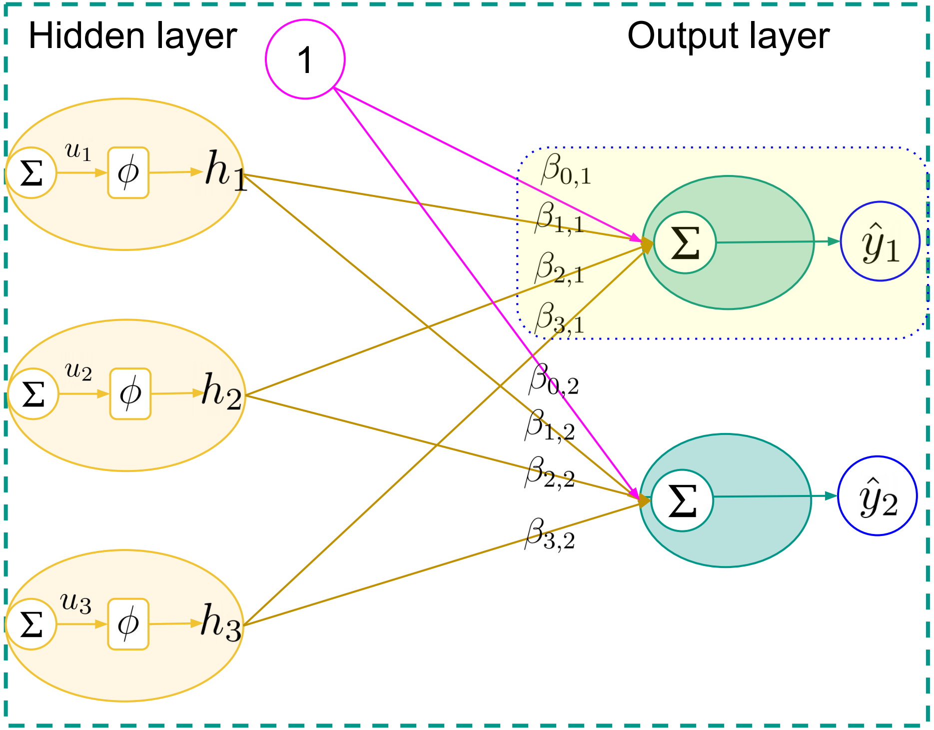
# Learning Algorithm



# Learning Algorithm



# Learning Algorithm




$$\hat{y}_{i,1} = \beta_{0,1} + \sum_{l=1}^{L} h_{i,l}\beta_{l,1}$$

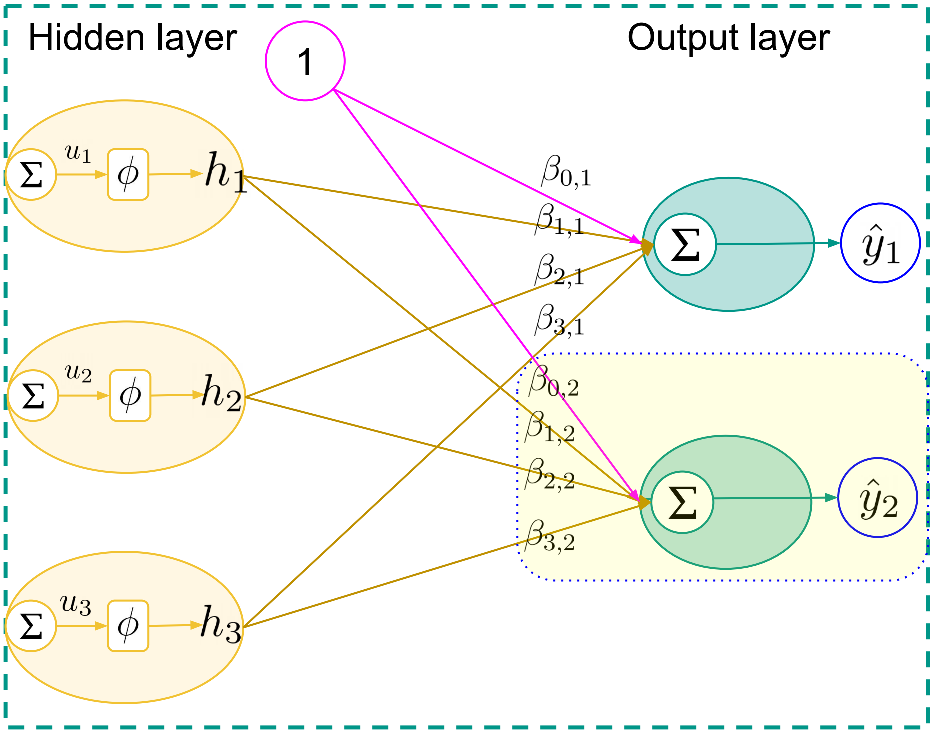
$$= \mathbf{h}_{(i,:)} \cdot \beta_{(:,1)}$$

$$= \begin{bmatrix} 1 & h_{i,1} & h_{i,2} & h_{i,3} \end{bmatrix} \cdot \begin{bmatrix} \beta_{0,1} \\ \beta_{1,1} \\ \beta_{2,1} \\ \beta_{3,1} \end{bmatrix}$$

$$\hat{y}_{(i,1)} = \mathbf{h}_{(i,:)} \cdot \beta_{(:,1)}$$

$i \in [1, \cdots, N]$

# Learning Algorithm

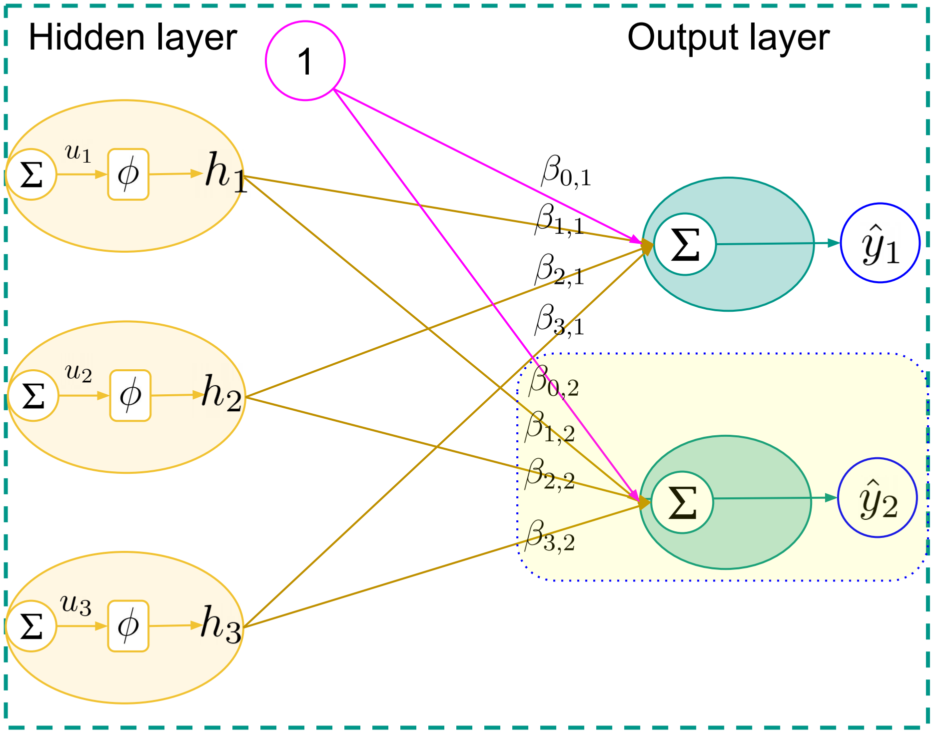



$$\begin{aligned}\hat{y}_{i,2} &= \beta_{0,2} + \sum_{l=1}^{L} h_{i,l}\beta_{l,2} \\ &= \mathbf{h}_{(i,:)} \cdot \beta_{(:,2)} \\ &= \begin{bmatrix} 1 & h_{i,1} & h_{i,2} & h_{i,3} \end{bmatrix} \cdot \begin{bmatrix} \beta_{0,2} \\ \beta_{1,2} \\ \beta_{2,2} \\ \beta_{3,2} \end{bmatrix}\end{aligned}$$

$$\hat{y}_{(i,2)} = \mathbf{h}_{(i,:)} \cdot \beta_{(:,2)}$$

$i \in [1, \cdots, N]$

# Learning Algorithm




$$\hat{\mathbf{Y}} = \mathbf{H} \cdot \boldsymbol{\beta}$$

$$\boldsymbol{\beta} = ?$$

# Generalized Pseudo inverse

Compute output weight using
the **Generalized Pseudo inverse**

$$\boxed{\mathbf{H}\boldsymbol{\beta} = \mathbf{Y}}$$

$$\mathbf{H}^\top \mathbf{H}\boldsymbol{\beta} = \mathbf{H}^\top \mathbf{Y}$$

$$(\mathbf{H}^\top \mathbf{H})^{-1}\mathbf{H}^\top \mathbf{H}\boldsymbol{\beta} = (\mathbf{H}^\top \mathbf{H})^{-1}\mathbf{H}^\top \mathbf{Y}$$

$$\mathbf{I}\boldsymbol{\beta} = (\mathbf{H}^\top \mathbf{H})^{-1}\mathbf{H}^\top \mathbf{Y}$$

numpy.linalg.pinv( ) $$\boxed{\boldsymbol{\beta} = (\mathbf{H}^\top \mathbf{H})^{-1}\mathbf{H}^\top \mathbf{Y}}$$

# Workshop