Back propagation

Tran Giang Son, tran-giang.son@usth.edu.vn

ICT Department, USTH

Neural Network

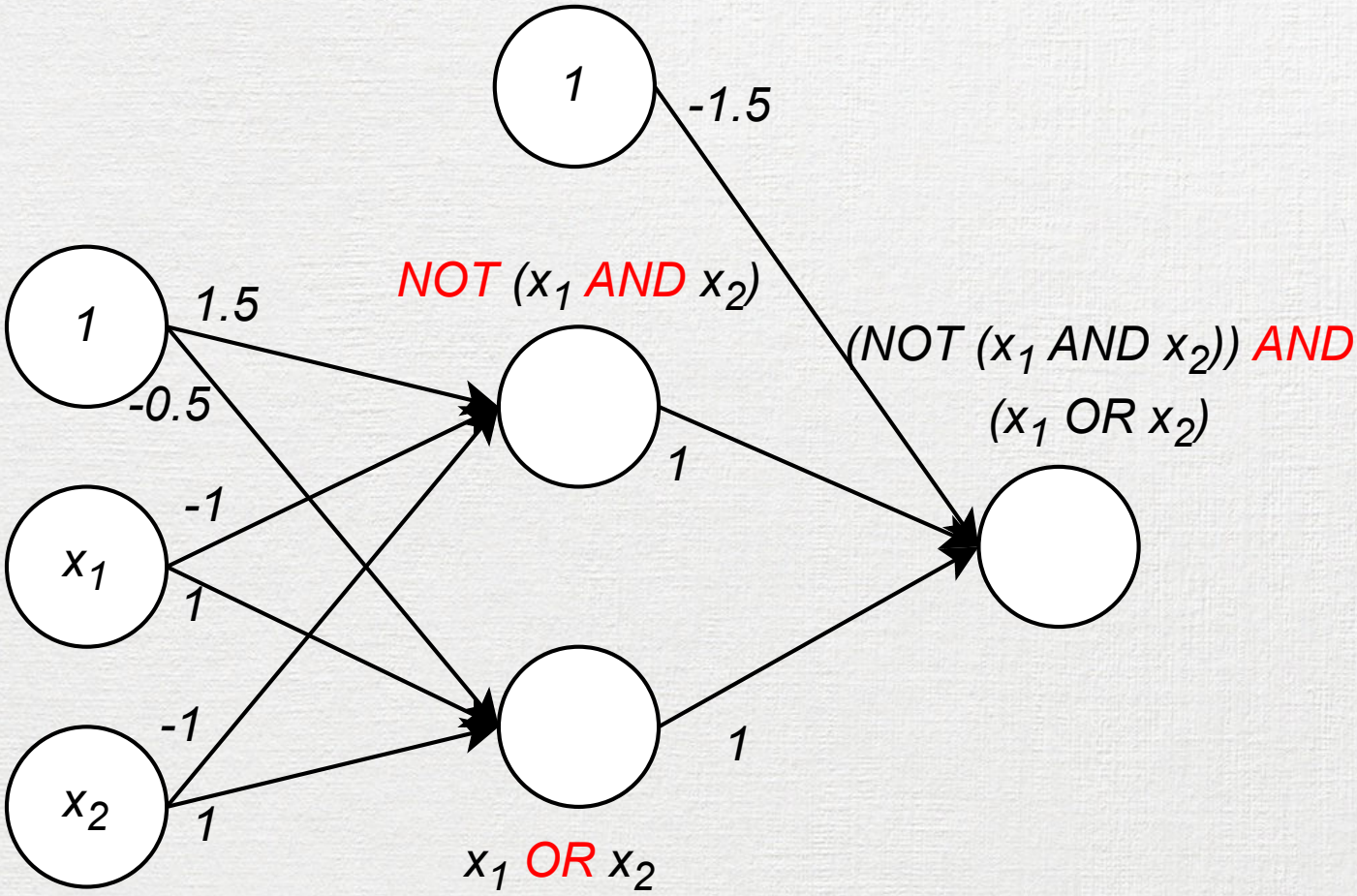
Neural Network: Remind XOR

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

Neural Network: Remind XOR

x_1	x_2	$x_1 \oplus x_2$	x_1 & x_2	!(x_1 & x_2)	$x_1 \mid x_2$	!(x_1 & x_2) & ($x_1 \mid x_2$)
0	0	0	0	1	0	0
0	1	1	0	1	1	1
1	0	1	0	1	1	1
1	1	0	1	0	1	0

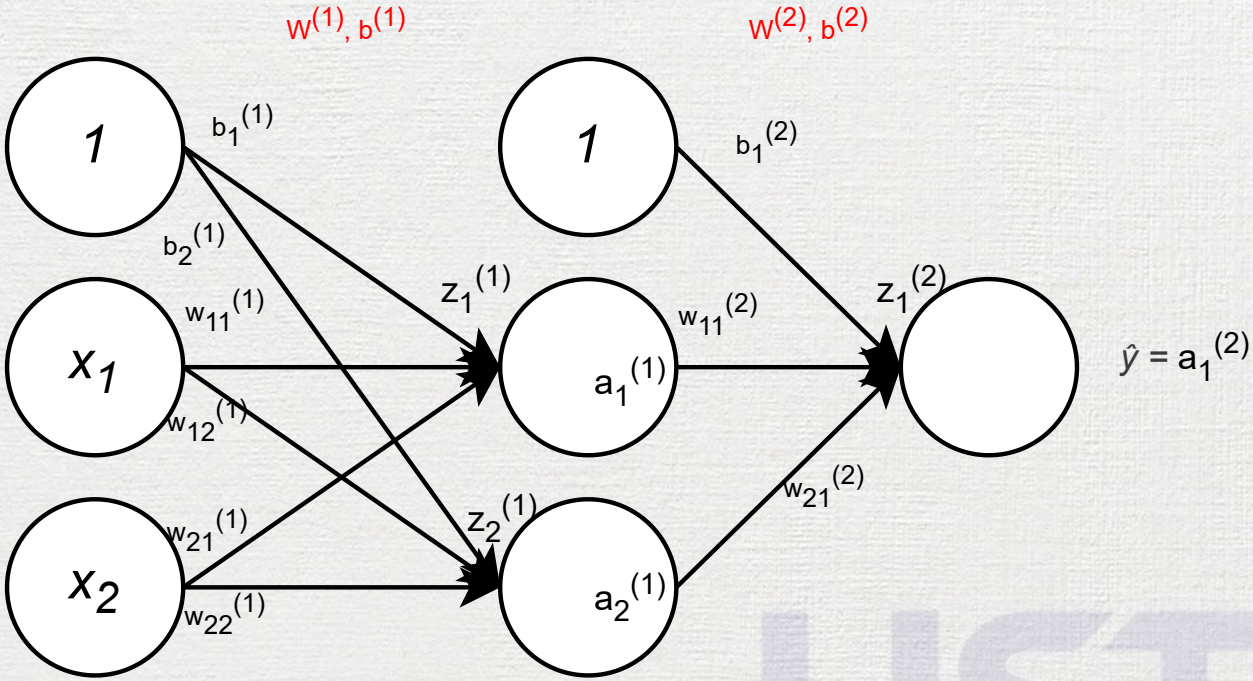
Neural Network: Remind XOR



$x_1 \oplus x_2$ with neural network

Neural Network Model for XOR

- Weights are *already known* for AND, OR.
  - How to apply for a new problem with new dataset without knowing the decompositions and operators?



Neural Network Model for XOR

- Model: 2-2-1:
  - 2 nodes in input layer
  - 2 nodes in hidden layer
  - 1 node in output layer
- Nodes 1 are added to calculate bias in next layers
- Each node in hidden layers and output layer are performed two steps
  (1) Linear sum
  (2) Apply activation function

Feedforward

Feedforward

- Single values:

$$
\begin{aligned}
z_1^{(1)} &= x_1 w_{11}^{(1)} + x_2 w_{21}^{(1)} + b_1^{(1)} \\
a_1^{(1)} &= \sigma(z_1^{(1)}) \\
z_2^{(1)} &= x_1 w_{12}^{(1)} + x_2 w_{22}^{(1)} + b_2^{(1)} \\
a_2^{(1)} &= \sigma(z_2^{(1)}) \\
z_1^{(2)} &= a_1^{(1)} w_{11}^{(2)} + a_2^{(1)} w_{21}^{(2)} + b_1^{(2)} \\
\hat{y} &= a_1^{(2)} = \sigma(z_1^{(2)})
\end{aligned}
\tag{1}
$$

Feedforward

- In matrix form:

$$
\begin{aligned}
Z^{(1)} &= X * W^{(1)} + b^{(1)} \\
A^{(1)} &= \sigma(Z^{(1)}) \\
Z^{(2)} &= A^{(1)} * W^{(2)} + b^{(2)} \\
\hat{Y} &= A^{(2)} = \sigma(Z^{(2)})
\end{aligned}
\tag{2}
$$

Dataset

Dataset XOR

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

Dataset XOR

- In matrix form:

$$X = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix}, Y = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \tag{3}$$

Loss Function

Loss Function

For each data point $(x^{[i]}, y_i)$, the loss function L is defined as follows:

$$L_i = -(y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)) \tag{4}$$

Where:

- y_i is the actual value
- $\hat{y}$ is the predicted value

$$\begin{aligned} \hat{y} = a_1^{(2)} &= \sigma(z_1^{(2)}) \\ &= \sigma(a_1^{(1)} w_{11}^{(2)} + a_2^{(1)} w_{21}^{(2)} + b_1^{(2)}) \end{aligned} \tag{5}$$

Loss Function

- For all data points, loss function J is defined as:

$$J = -\frac{1}{N}\sum_{i=1}^{N}(y_i \log(\hat{y}_i) + (1 - y_i) * log(1 - \hat{y}_i)) \tag{6}$$

Gradient Descent

- For each neuron in each layer, optimize $w_{ij}^{(k)}$ using gradient descend
  - Similar to logistic regression in labwork 3 :)

Practice!

Labwork 5: Neural Network with Backpropagation

- Copy your code from labwork 4 into labwork 5
- Implement **(from scratch!)** back propagation with gradient descend
  - Input: a csv file for training data
  - Output: a properly trained neural network, with weights and bias optimized using back propagation
- Write a report (in LaTeX):
  - Name it « Report.5.Backprop.tex »
  - How you implement back propagation
  - Experiment with the XOR dataset and the house price-size dataset.
- Push your code and report to your forked repository