2-1 Neural Network with One Hidden Layer I

(one training example)

Zhonglei Wang

WISE and SOE, XMU, 2025

Contents

Intuition

1. Illustrate basic concepts using only ONE training example $(\boldsymbol{x}, y)$
2. A neural network can be viewed as a function of features $\boldsymbol{x}$ with parameter $\boldsymbol{\theta}$
3. To obtain an estimator of the model parameter $\boldsymbol{\theta}$, we use a (batch) gradient descent algorithm
4. An essential step is

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \alpha \frac{\partial \mathcal{J}}{\partial \boldsymbol{\theta}} \left(\boldsymbol{\theta}^{(t)} \right)$$

   - $\boldsymbol{\theta}^{(t)}$: current model parameter
   - What are the cost function and its partial derivatives?

Intuition

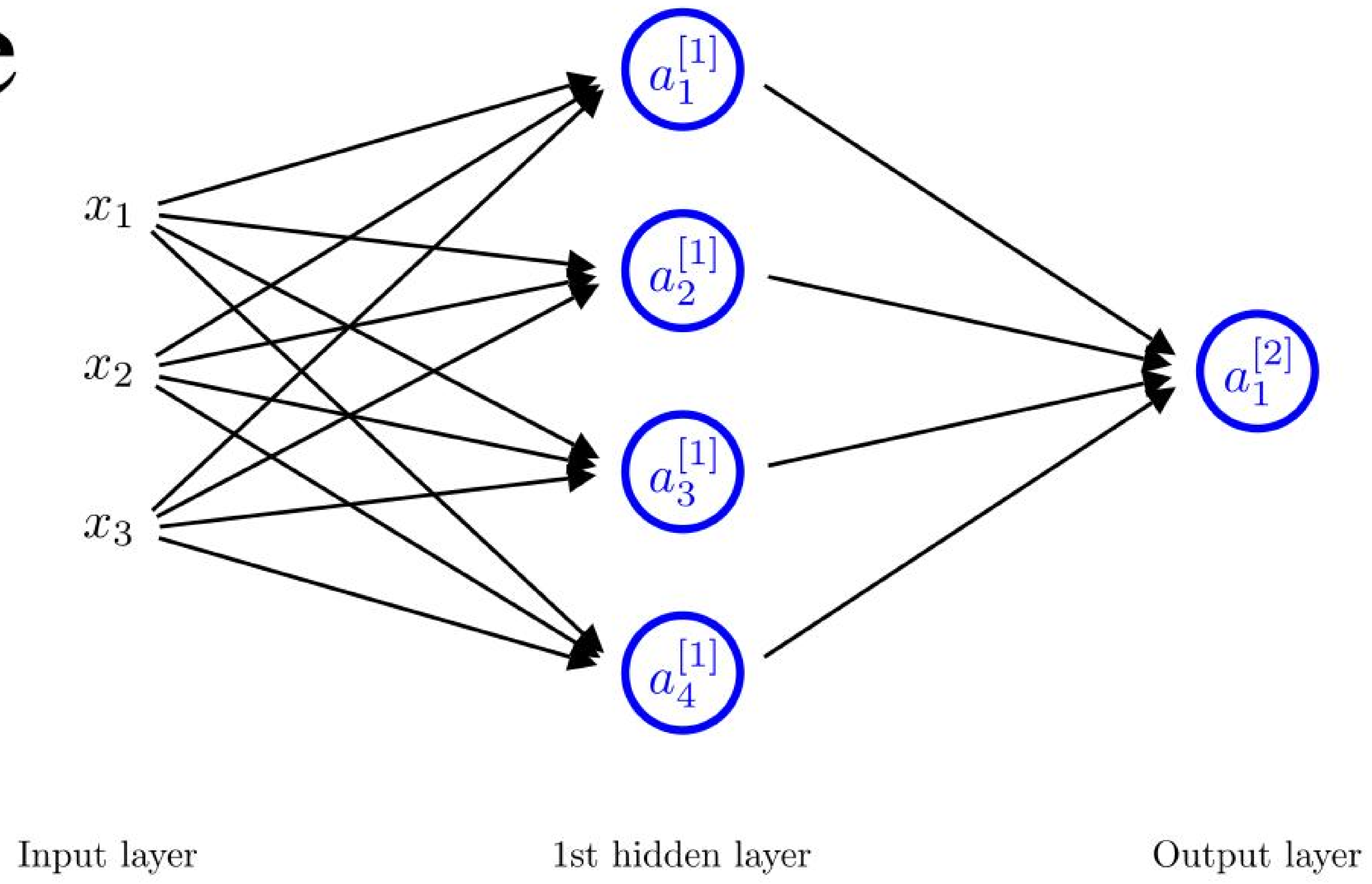
1. To obtain $\partial \mathcal{J}(\boldsymbol{\theta}^{(t)})/\partial \boldsymbol{\theta}$, we introduce
   - Forward propagation: based on the current parameter, calculate the “activated” values as well as the cost function
   - Backpropagation: based on “those” values, obtain partial derivatives
2. We use a toy neural network to introduce forward propagation and backpropagation.

Revisit logistic regression

1. The model is too simple in practice
2. Why not use more “circles”?

Neural network example

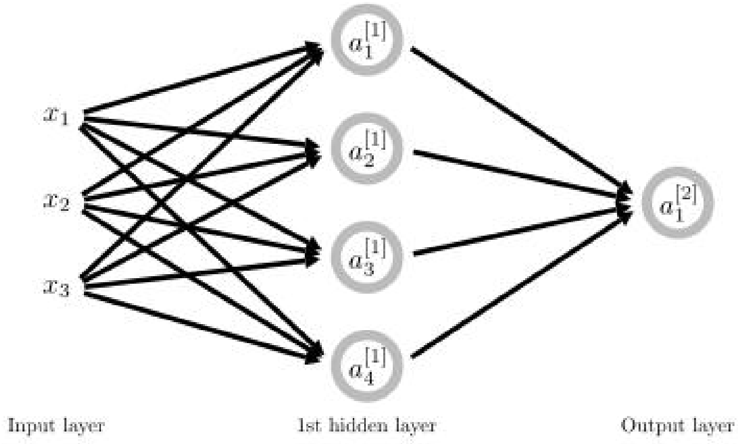
1. Assume $\boldsymbol{x} = (x_1, x_2, x_3)^{\mathrm{T}}$



2. Each circle represents two operations
   - Linear transformation with two model parameters, including a bias and a weight
   - Activation (nonlinear transformation) without model parameters

Neural network example

1. Remarks
   - Different model parameters are used for different “circles” to extract different information
   - In other words, we “contruct” a set of “new” features in the hidden layer
   - Compared with logistic regression model, this neural network uses one hidden layer to extracts more information
   - For the output layer, the hidden layer can be regarded as its “input” layer



Calculation	Parameters
$z_1^{[1]} = b_1^{[1]} + \boldsymbol{x}^{\mathrm{T}} \boldsymbol{w}_1^{[1]}, \quad a_1^{[1]} = \sigma(z_1^{[1]})$	$b_1^{[1]} \quad \boldsymbol{w}_1^{[1]}$
$z_2^{[1]} = b_2^{[1]} + \boldsymbol{x}^{\mathrm{T}} \boldsymbol{w}_2^{[1]}, \quad a_2^{[1]} = \sigma(z_2^{[1]})$	$b_2^{[1]} \quad \boldsymbol{w}_2^{[1]}$
$z_3^{[1]} = b_3^{[1]} + \boldsymbol{x}^{\mathrm{T}} \boldsymbol{w}_3^{[1]}, \quad a_3^{[1]} = \sigma(z_3^{[1]})$	$b_3^{[1]} \quad \boldsymbol{w}_3^{[1]}$
$z_4^{[1]} = b_4^{[1]} + \boldsymbol{x}^{\mathrm{T}} \boldsymbol{w}_4^{[1]}, \quad a_4^{[1]} = \sigma(z_4^{[1]})$	$b_4^{[1]} \quad \boldsymbol{w}_4^{[1]}$
$z_1^{[2]} = b_1^{[2]} + (\boldsymbol{a}^{[1]})^{\mathrm{T}} \boldsymbol{w}_1^{[2]}, \quad a_1^{[2]} = \sigma(z_1^{[2]})$	$b_1^{[2]} \quad \boldsymbol{w}_1^{[2]}$

Vectorization

1. Denote
   - L : number of layers in the neural network
   - $d^{[l]}$: number of neurons in the lth layer $\quad (l = 0, \ldots, L)$
   - $\boldsymbol{a}^{[l]} = (a_1^{[l]}, \ldots, a_{d^{[l]}}^{[l]})^{\mathrm{T}} \in \mathbb{R}^{d^{[l]} \times 1}$
   - $\boldsymbol{W}^{[l]} = (\boldsymbol{w}_1^{[l]}, \ldots, \boldsymbol{w}_{d^{[l]}}^{[l]})^{\mathrm{T}} \in \mathbb{R}^{d^{[l]} \times d^{[l-1]}}$
   - $\boldsymbol{b}^{[l]} = (b_1^{[l]}, \ldots, b_{d^{[l]}}^{[l]})^{\mathrm{T}} \in \mathbb{R}^{d^{[l]} \times 1}$
2. Model parameters
   - $\{(\boldsymbol{b}^{[l]}, \boldsymbol{W}^{[l]}) : l = 1, \ldots, L\}$

Vectorization

1. For the previous example, we have
   - $d^{[0]} = 3, d^{[1]} = 4, d^{[2]} = 1$
   - $\boldsymbol{W}^{[1]} \in \mathbb{R}^{4\times 3}, \boldsymbol{W}^{[2]} \in \mathbb{R}^{1\times 4}$
   - $\boldsymbol{b}^{[1]} \in \mathbb{R}^{4\times 1}, b^{[2]} \in \mathbb{R}^{1\times 1}$

Forward propagation

1. Forward propagation obtains “activated” values using the CURRENT parameters
2. Assume the current model parameters are $\boldsymbol{b}^{[1]}, \boldsymbol{W}^{[1]}, b^{[2]}, \boldsymbol{W}^{[2]}$
3. The calculation can be simpified as

$$\boldsymbol{z}^{[1]} = \boldsymbol{b}^{[1]} + \boldsymbol{W}^{[1]}\boldsymbol{x}$$
$$\boldsymbol{a}^{[1]} = \sigma(\boldsymbol{z}^{[1]})$$
$$z^{[2]} = \boldsymbol{b}^{[2]} + \boldsymbol{W}^{[2]}\boldsymbol{a}^{[1]}$$
$$a^{[2]} = \sigma(z^{[2]})$$

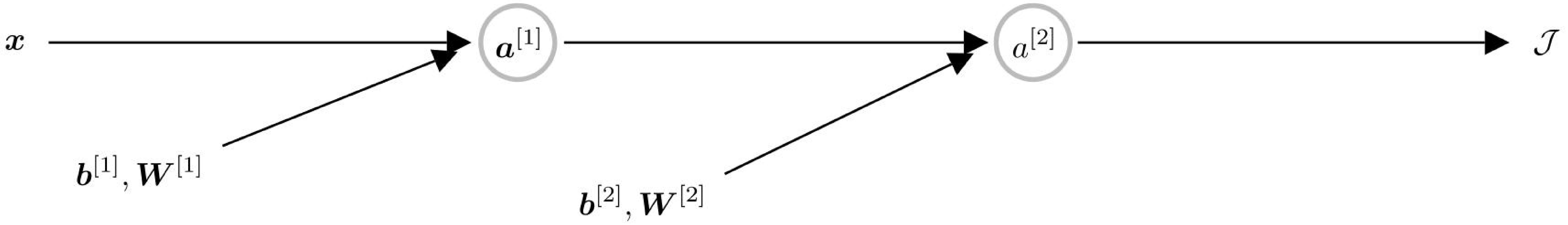
Forward propagation

1. Consider a binary response: $y \in \{0, 1\}$
2. $a^{[2]}$ is the estimated value for $P(y = 1 \mid \boldsymbol{x})$
3. Thus, we consider a loss function

$$\mathcal{J} = \mathcal{L} = -\left\{ y \log a^{[2]} + (1 - y) \log \left(1 - a^{[2]} \right) \right\}$$

Forward propagation

1. Visualize the calculation details



Forward propagation

$$\mathcal{J} = -\left\{ y \log a^{[2]} + (1-y) \log \left(1 - a^{[2]}\right) \right\}$$

$$\boldsymbol{z}^{[1]} = \boldsymbol{b}^{[1]} + \boldsymbol{W}^{[1]} \boldsymbol{x}$$
$$\boldsymbol{a}^{[1]} = \sigma(\boldsymbol{z}^{[1]})$$

$$z^{[2]} = b^{[2]} + \boldsymbol{W}^{[2]} \boldsymbol{a}^{[1]}$$
$$a^{[2]} = \sigma(z^{[2]})$$

Backpropagation

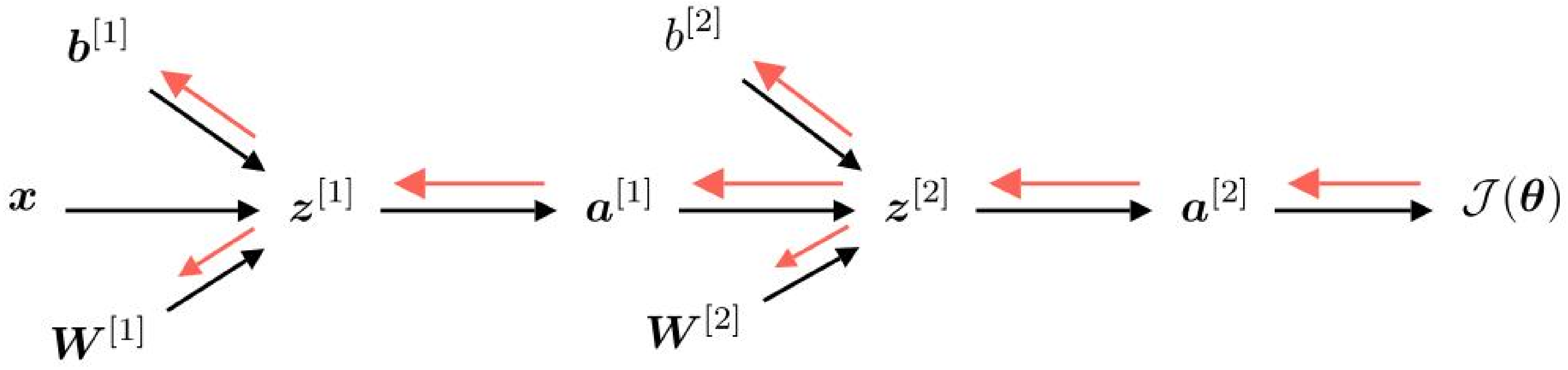
1. Based on the current model parameters, backpropagation is to obtain

$$\frac{\partial \mathcal{J}}{\partial \boldsymbol{b}^{[l]}}, \quad \frac{\partial \mathcal{J}}{\partial \boldsymbol{W}^{[l]}} \quad (l = 1, \ldots, L)$$

2. Core technique: chain rule

    - Denote $f(\boldsymbol{x}) = g \circ h(\boldsymbol{x})$
    - Then, we have

$$\frac{\partial f}{\partial \boldsymbol{x}} = \frac{\partial g}{\partial h} \cdot \frac{\partial h}{\partial \boldsymbol{x}}$$



$$\frac{\partial \mathcal{J}(\boldsymbol{\theta})}{\partial b^{[2]}} = a^{[2]} - y \quad \frac{\partial \mathcal{J}(\boldsymbol{\theta})}{\partial \boldsymbol{W}^{[2]}} = (a^{[2]} - y)(\boldsymbol{a}^{[1]})^{\mathrm{T}}$$

$$\frac{\partial \mathcal{J}(\boldsymbol{\theta})}{\partial \boldsymbol{b}^{[1]}} = (a^{[2]} - y)\boldsymbol{D}(\boldsymbol{W}^{[2]})^{\mathrm{T}} \qquad \frac{\partial \mathcal{J}(\boldsymbol{\theta})}{\partial \boldsymbol{W}^{[1]}} = (a^{[2]} - y)\boldsymbol{D}(\boldsymbol{W}^{[2]})^{\mathrm{T}}\boldsymbol{x}^{\mathrm{T}}$$

$$\boldsymbol{D} = \mathrm{diag}(\{a_j^{[1]}(1 - a_j^{[1]}) : j = 1, \ldots, d^{[1]}\})$$

We only need to cache $\boldsymbol{a}^{[1]}$ and $a^{[2]}$

(Batch) gradient descent algorithm

Step1. Randomly initialize $\boldsymbol{\theta}^{(0)}$

Step2. Based on $\boldsymbol{\theta}^{(t)}$ obtain

$$\nabla \mathcal{J}\left(\boldsymbol{\theta}^{(t)}\right) = \frac{\partial \mathcal{J}}{\partial \boldsymbol{\theta}}(\boldsymbol{\theta}^{(t)})$$

Step3. Update parameter

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \alpha \nabla \mathcal{J}\left(\boldsymbol{\theta}^{(t)}\right)$$

Step4. Go back to Step 2 until convergence