AI and Deep Learning

2-1 Neural Network with One Hidden Layer I

(one training example)

Zhonglei Wang

WISE, SOE, CHOW

XMU

(Version v1)

Contents

1. Revisit logistic regression

2. Forward propagation

3. Backpropagation

4. (Batch) Gradient descent algorithm

Background

1. Illustrate basic concepts using only ONE training example $(\boldsymbol{x}, y)$ $\quad (y \in \{0, 1\})$
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

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

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

Dimension of partial derivatives

1. Consider $\boldsymbol{f}(\boldsymbol{x})$,
   - $\boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), \ldots, f_k(\boldsymbol{x}))^{\mathrm{T}}$: Column vector of length k
   - $\boldsymbol{x} = (x_1, \ldots, x_m)^{\mathrm{T}}$: Column vector of length m
2. In this class, we define

$$\frac{\partial \boldsymbol{f}^{\mathrm{T}}}{\partial \boldsymbol{x}} = \left(\frac{\partial f_1}{\partial \boldsymbol{x}}, \ldots, \frac{\partial f_k}{\partial \boldsymbol{x}} \right) = \begin{pmatrix} \partial f_1(\boldsymbol{x})/\partial x_1 & \ldots & \partial f_k(\boldsymbol{x})/\partial x_1 \\ \partial f_1(\boldsymbol{x})/\partial x_2 & \ldots & \partial f_k(\boldsymbol{x})/\partial x_2 \\ \vdots & \ddots & \vdots \\ \partial f_1(\boldsymbol{x})/\partial x_m & \ldots & \partial f_k(\boldsymbol{x})/\partial x_m \end{pmatrix} \in \mathbb{R}^{m \times k}$$

   - To make the dimension clear, we use a transpose in the numerator or denominator
   - Make $\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \alpha \cdot \partial \mathcal{J} / \partial \boldsymbol{\theta} \left(\boldsymbol{\theta}^{(t)} \right)$ mathematically rigorous ($k = 1$)

Dimension of partial derivatives

1. In advanced math classes, we define derivatives to facilitate linear approximation ($\boldsymbol{\epsilon} \approx 0$)

$$\boldsymbol{f}(\boldsymbol{x}+\boldsymbol{\epsilon}) = \begin{pmatrix} f_1(\boldsymbol{x}+\boldsymbol{\epsilon}) \\ f_2(\boldsymbol{x}+\boldsymbol{\epsilon}) \\ \vdots \\ f_k(\boldsymbol{x}+\boldsymbol{\epsilon}) \end{pmatrix} \approx \begin{pmatrix} f_1(\boldsymbol{x}) \\ f_2(\boldsymbol{x}) \\ \vdots \\ f_k(\boldsymbol{x}) \end{pmatrix} + \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} \cdot \boldsymbol{\epsilon}$$

- $\partial \boldsymbol{f}/\partial \boldsymbol{x}$: Transpose of the one we defined in the previous slide

$$\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} = \begin{pmatrix} \partial f_1(\boldsymbol{x})/\partial x_1 & \dots & \partial f_1(\boldsymbol{x})/\partial x_m \\ \partial f_2(\boldsymbol{x})/\partial x_1 & \dots & \partial f_2(\boldsymbol{x})/\partial x_m \\ \vdots & \ddots & \vdots \\ \partial f_k(\boldsymbol{x})/\partial x_1 & \dots & \partial f_k(\boldsymbol{x})/\partial x_m \end{pmatrix} \in \mathbb{R}^{k \times m}$$

Dimension of partial derivatives

1. For the convenience of our discussion, we DO NOT adopt the form of gradient in those advanced mathematics

Examples

1. Example 1: Consider $f(\boldsymbol{x})$
   - $\boldsymbol{x} = (x_1, \ldots, x_m)^{\mathrm{T}}$: Column vector
   - $f(\boldsymbol{x})$: Univariate function
2. Then, we have

$$\frac{\partial f}{\partial \boldsymbol{x}} = \left(\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_m} \right)^{\mathrm{T}}$$

3. $\partial f / \partial \boldsymbol{x}$ has the same dimension with $\boldsymbol{x}$

Examples

1. Example 2: Consider $f(\boldsymbol{x})$
   - $\boldsymbol{x} = (x_1, \ldots, x_m)$: Row vector
   - $f(\boldsymbol{x})$: Univariate function
2. Then, we have

$$\frac{\partial f}{\partial \boldsymbol{x}} = \left(\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_m} \right)$$

3. $\partial f / \partial \boldsymbol{x}$ has the same dimension with $\boldsymbol{x}$

Examples

1. Consider $f(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{x}^{\mathrm{T}} \cdot \boldsymbol{y} = \sum_{j=1}^{m} x_j y_j$
   - $\boldsymbol{x} = (x_1, \ldots, x_m)^{\mathrm{T}}$
   - $\boldsymbol{y} = (y_1, \ldots, y_m)^{\mathrm{T}}$
2. Then, we have

$$\frac{\partial f}{\partial \boldsymbol{x}} = \boldsymbol{y}, \qquad \frac{\partial f}{\partial \boldsymbol{y}} = \boldsymbol{x},$$

$$\frac{\partial f}{\partial \boldsymbol{x}^{\mathrm{T}}} = \boldsymbol{y}^{\mathrm{T}}, \qquad \frac{\partial f}{\partial \boldsymbol{y}^{\mathrm{T}}} = \boldsymbol{x}^{\mathrm{T}}$$

Examples

1. Example 3: Consider $f(\boldsymbol{X})$
   - $\boldsymbol{X} = (x_{ij}) \in \mathbb{R}^{m \times k}$: $m \times k$ matrix
   - $f(\boldsymbol{X})$: Univariate function
2. Then, we have

$$\frac{\partial f}{\partial \boldsymbol{X}} = \left(\frac{\partial f}{\partial x_{ij}} \right)$$

3. $\partial f / \partial \boldsymbol{X}$ has the same dimension with $\boldsymbol{X}$

Examples

1. Example 4 (chain rule): Consider

$$f(\boldsymbol{x}) = g \circ \boldsymbol{h}(\boldsymbol{x})$$

- $\boldsymbol{x}$: Scalar, column vector, row vector or matrix
- $f(\boldsymbol{x})$: Univariate function
- $\boldsymbol{h}(\boldsymbol{x}) = (h_1(\boldsymbol{x}), \ldots, h_m(\boldsymbol{x}))^{\mathrm{T}}$

2. Then, we have

$$\frac{\partial f}{\partial \boldsymbol{x}} = \sum_{j=1}^{m} \frac{\partial h_j}{\partial \boldsymbol{x}} \cdot \frac{\partial g}{\partial h_j}$$

Examples

1. If $\boldsymbol{x}$ is a scalar or a column vector, we further have

$$\frac{\partial f}{\partial \boldsymbol{x}} = \sum_{j=1}^{m} \frac{\partial h_j}{\partial \boldsymbol{x}} \cdot \frac{\partial g}{\partial h_j} = \left(\frac{\partial h_1}{\partial \boldsymbol{x}}, \ldots, \frac{\partial h_m}{\partial \boldsymbol{x}} \right) \cdot \begin{pmatrix} \frac{\partial g}{\partial h_1} \\ \vdots \\ \frac{\partial g}{\partial h_m} \end{pmatrix} = \frac{\partial \boldsymbol{h}^{\mathrm{T}}}{\partial \boldsymbol{x}} \cdot \frac{\partial g}{\partial \boldsymbol{h}}$$

   - $\partial g / \partial \boldsymbol{h} = (\partial g / \partial h_1, \ldots, \partial g / \partial h_m)^{\mathrm{T}}$
   - $\partial \boldsymbol{h}^{\mathrm{T}} / \partial \boldsymbol{x} = (\partial h_1 / \partial \boldsymbol{x}, \ldots, \partial h_m / \partial \boldsymbol{x})$

2. That is, the calculation can be vectorized
3. However, the essential part is to get $\partial h_j / \partial \boldsymbol{x}$
4. Vectorization may fail if $\boldsymbol{x}$ is a matrix
5. Question: How about the case when $\boldsymbol{x}$ is a row vector?

Goal

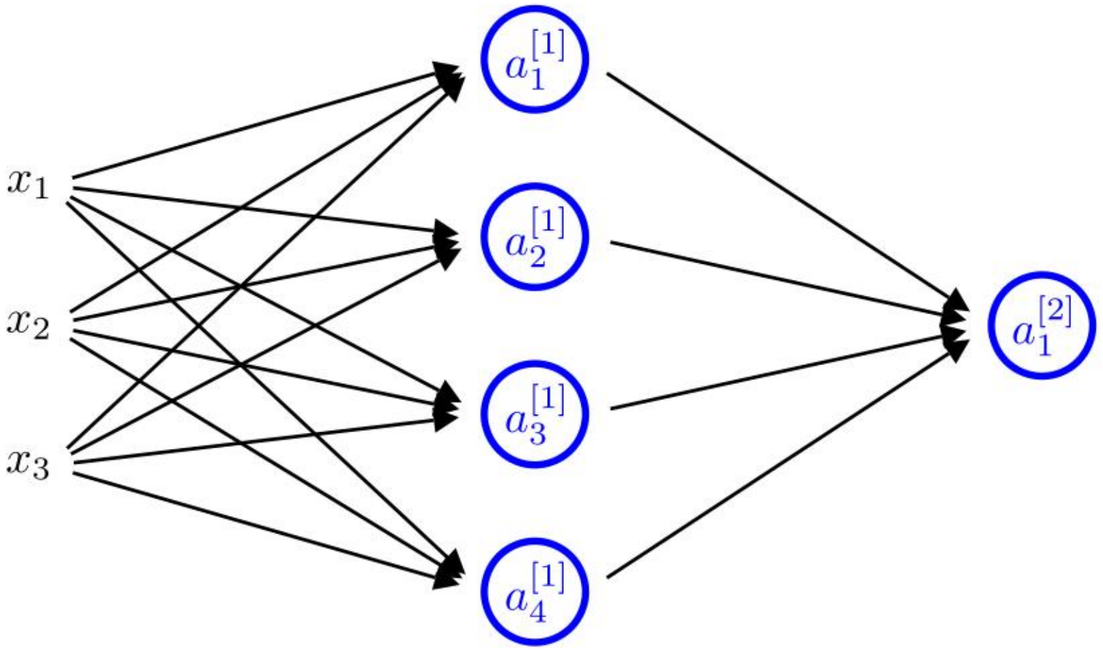
1. To obtain $\partial \mathcal{J}(\boldsymbol{\theta}^{(t)})/\partial \boldsymbol{\theta}$, we introduce
   - Forward propagation: Based on the current parameter, calculate the "activated" values as well as the cost function
   - Backpropagation: Based on "those" values, obtain partial derivatives
2. 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 $\boldsymbol{x} = (x_1, x_2, x_3)^{\mathrm{T}}$



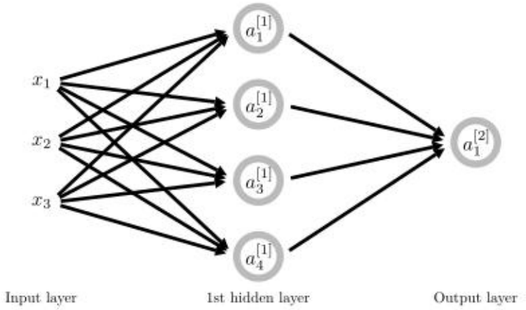
2. Each circle represents two operations:

- 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
- For the output layer, the hidden layer can be regarded as its “input” layer
- Compared with a logistic regression model, this neural network uses one hidden layer to extracts more information



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

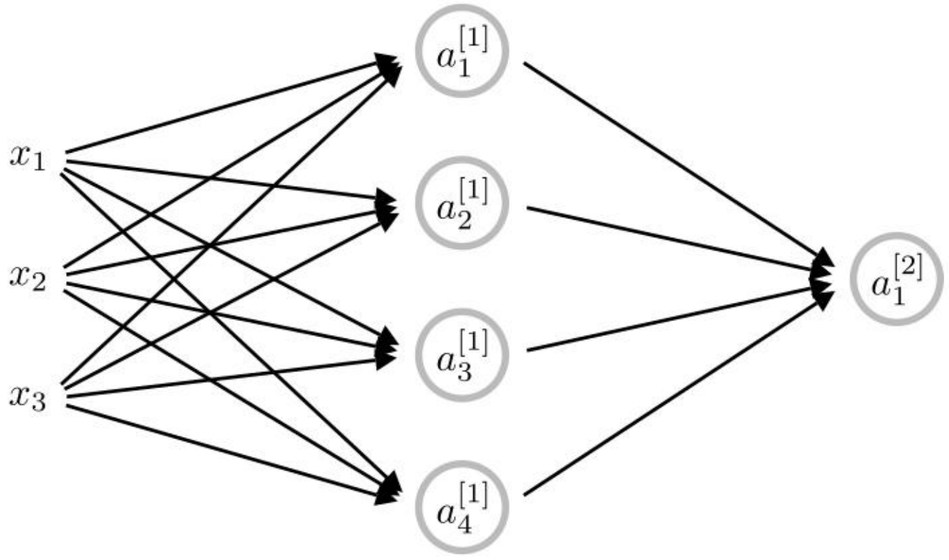
Vectorization

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

- $d^{[0]} = 3, d^{[1]} = 4, d^{[2]} = 1$
- $\boldsymbol{W}^{[1]} = (\boldsymbol{w}_1^{[1]}, \boldsymbol{w}_2^{[1]}, \boldsymbol{w}_3^{[1]}, \boldsymbol{w}_4^{[1]})^{\mathrm{T}} \in \mathbb{R}^{4\times 3}, \boldsymbol{W}^{[2]} = (\boldsymbol{w}_1^{[2]})^{\mathrm{T}} \in \mathbb{R}^{1\times 4}$
- $\boldsymbol{b}^{[1]} = (b_1^{[1]}, b_2^{[1]}, b_3^{[1]}, b_4^{[1]})^{\mathrm{T}} \in \mathbb{R}^{4\times 1}, b^{[2]} \in \mathbb{R}$



Forward propagation

1. Forward propagation obtains "activated" values using 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

$$\begin{aligned}
\boldsymbol{z}^{[1]} &= \boldsymbol{b}^{[1]} + \boldsymbol{W}^{[1]} \cdot \boldsymbol{x}, \\
\boldsymbol{a}^{[1]} &= \sigma(\boldsymbol{z}^{[1]}), \\
z^{[2]} &= b^{[2]} + \boldsymbol{W}^{[2]} \cdot \boldsymbol{a}^{[1]}, \\
a^{[2]} &= \sigma(z^{[2]})
\end{aligned}$$

$$\boldsymbol{z}^{[1]} = \begin{pmatrix} b_1^{[1]} \\ b_2^{[1]} \\ b_3^{[1]} \\ b_4^{[1]} \end{pmatrix} + \begin{pmatrix} (\boldsymbol{w}_1^{[1]})^{\mathrm{T}} \\ (\boldsymbol{w}_2^{[1]})^{\mathrm{T}} \\ (\boldsymbol{w}_3^{[1]})^{\mathrm{T}} \\ (\boldsymbol{w}_4^{[1]})^{\mathrm{T}} \end{pmatrix} \cdot \boldsymbol{x} = \begin{pmatrix} b_1^{[1]} + (\boldsymbol{w}_1^{[1]})^{\mathrm{T}} \cdot \boldsymbol{x} \\ b_2^{[1]} + (\boldsymbol{w}_2^{[1]})^{\mathrm{T}} \cdot \boldsymbol{x} \\ b_3^{[1]} + (\boldsymbol{w}_3^{[1]})^{\mathrm{T}} \cdot \boldsymbol{x} \\ b_4^{[1]} + (\boldsymbol{w}_4^{[1]})^{\mathrm{T}} \cdot \boldsymbol{x} \end{pmatrix}$$

Forward propagation

1. Forward propagation obtains "activated" values using 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

$$\boldsymbol{z}^{[1]} = \boldsymbol{b}^{[1]} + \boldsymbol{W}^{[1]} \cdot \boldsymbol{x},$$

$$\boldsymbol{a}^{[1]} = \sigma(\boldsymbol{z}^{[1]}),$$

$$z^{[2]} = b^{[2]} + \boldsymbol{W}^{[2]} \cdot \boldsymbol{a}^{[1]},$$

$$a^{[2]} = \sigma(z^{[2]})$$

$$\boldsymbol{a}^{[1]} = \begin{pmatrix} \sigma(z_1^{[1]}) \\ \sigma(z_2^{[1]}) \\ \sigma(z_3^{[1]}) \\ \sigma(z_4^{[1]}) \end{pmatrix}$$

Forward propagation

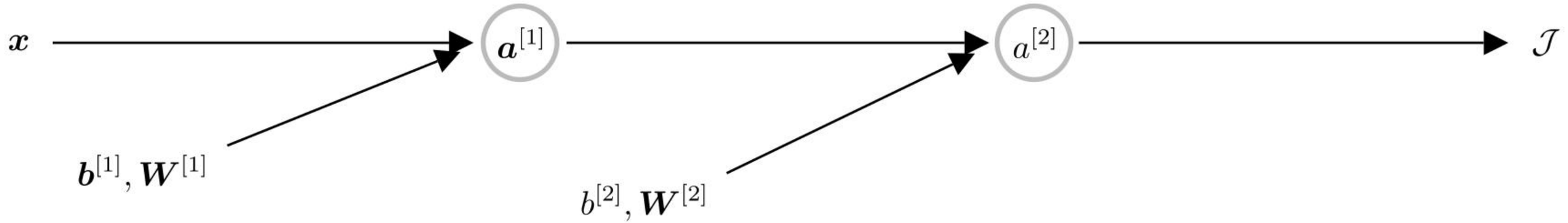
1. Consider a binary response ($y \in \{0, 1\}$)
2. $a^{[2]}$ is the estimated value for $P(y = 1 \mid \boldsymbol{x})$
3. Thus, we consider a loss function

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

4. Question: What would $a^{[2]}$ and $\mathcal{L}$ be if $y \in \mathbb{R}$?
5. Question: What about the case when the range of labels is nonnegative?

Forward propagation

1. Visualize the calculation details



Forward propagation

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

$$\boldsymbol{z}^{[1]} = \boldsymbol{b}^{[1]} + \boldsymbol{W}^{[1]} \cdot \boldsymbol{x}$$

$$\boldsymbol{a}^{[1]} = \sigma(\boldsymbol{z}^{[1]})$$

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

$$a^{[2]} = \sigma(z^{[2]})$$

Remarks

1. When obtaining $\boldsymbol{a}^{[1]}$ in the first layer, $\boldsymbol{x}$ is the input
2. When obtaining $a^{[2]}$ in the second layer, $\boldsymbol{a}^{[1]}$ can be treated as the input
3. That is, when obtaining activated values in the lth layer, those in the previous layer can be treated as the input
4. This is the general case for forward propagation, so forward propagation can be coded by a for-loop

Backpropagation

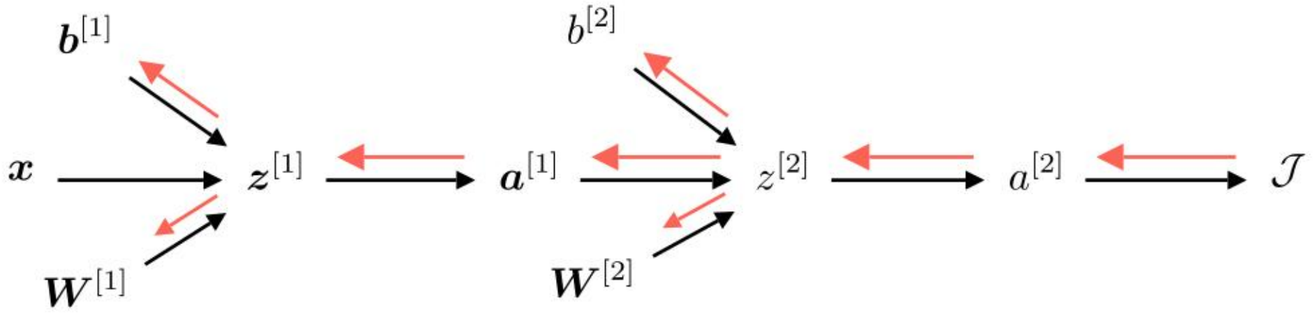
1. Based on current model parameters, backpropagation is to obtain

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

2. Core technique: Chain rule

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

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

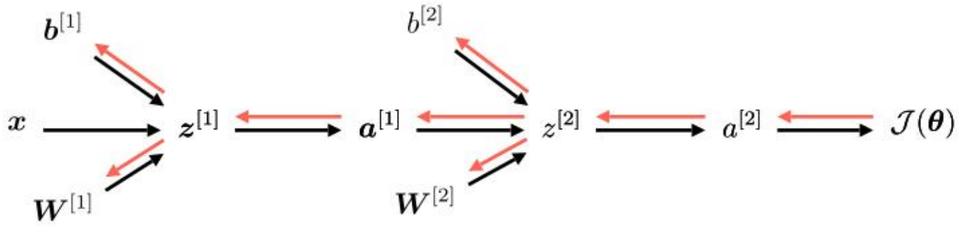


$$\frac{\partial \mathcal{J}}{\partial b^{[2]}} = a^{[2]} - y \qquad \frac{\partial \mathcal{J}}{\partial \boldsymbol{W}^{[2]}} = (a^{[2]} - y) \cdot (\boldsymbol{a}^{[1]})^{\mathrm{T}}$$

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

1. $\boldsymbol{D} = \mathrm{diag}(\{a_j^{[1]}(1 - a_j^{[1]}) : j = 1, \ldots, d^{[1]}\})$
2. We only need to cache $\boldsymbol{a}^{[1]}$ and $a^{[2]}$

Remarks

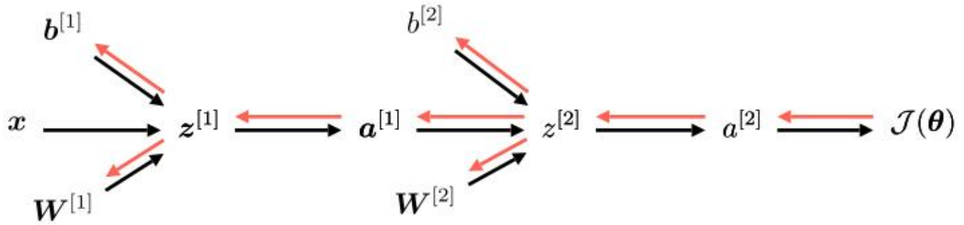


1. The form of $\partial \mathcal{J}/\partial a^{[2]}$ is determined by the form of the cost function $\mathcal{J}$
   - We have shown the result for binary classification problems
   - Derive $\partial \mathcal{J}/\partial a^{[2]}$ if $\mathcal{J} = (y - a^{[2]})^2/2$ for regression problems by yourself
2. Once we obtain $\partial \mathcal{J}/\partial a^{[2]}$, we use a chain rule to get

$$\frac{\partial \mathcal{J}}{\partial z^{[2]}} = \frac{\partial a^{[2]}}{\partial z^{[2]}} \cdot \frac{\partial \mathcal{J}}{\partial a^{[2]}}$$

   - Result for $\partial a^{[2]}/\partial z^{[2]}$ is determined by the form of activation function to get $a^{[2]}$
   - $\partial \mathcal{J}/\partial z^{[2]} = a^{[2]} - y$ for binary classification problems

Remarks



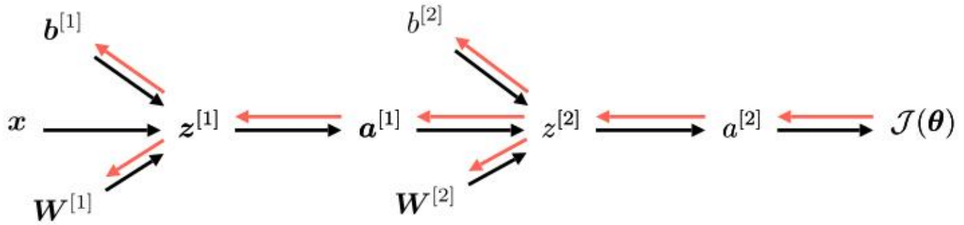
1. Once we obtain $\partial \mathcal{J} / \partial z^{[2]}$, we can easily get

$$\frac{\partial \mathcal{J}}{\partial b^{[2]}} = \frac{\partial z^{[2]}}{\partial b^{[2]}} \cdot \frac{\partial \mathcal{J}}{\partial z^{[2]}} = \frac{\partial \mathcal{J}}{\partial z^{[2]}},$$

$$\frac{\partial \mathcal{J}}{\partial \boldsymbol{W}^{[2]}} = \frac{\partial z^{[2]}}{\partial \boldsymbol{W}^{[2]}} \cdot \frac{\partial \mathcal{J}}{\partial z^{[2]}} = (\boldsymbol{a}^{[1]})^{\mathrm{T}} \cdot \frac{\partial \mathcal{J}}{\partial z^{[2]}}$$

2. That is, to obtain derivatives with respect to the model parameters in the second layer, it is desirable to obtain $\partial \mathcal{J} / \partial z^{[2]}$
3. Similar argument applies to the case for the first layer

Remarks



1. Once we have $\partial \mathcal{J}/\partial \boldsymbol{a}^{[1]}$, we can easily get

$$\frac{\partial \mathcal{J}}{\partial \boldsymbol{z}^{[1]}} = \frac{\partial (\boldsymbol{a}^{[1]})^{\mathrm{T}}}{\partial \boldsymbol{z}^{[1]}} \cdot \frac{\partial \mathcal{J}}{\partial \boldsymbol{a}^{[1]}} = \boldsymbol{D} \cdot \frac{\partial \mathcal{J}}{\partial \boldsymbol{a}^{[1]}},$$

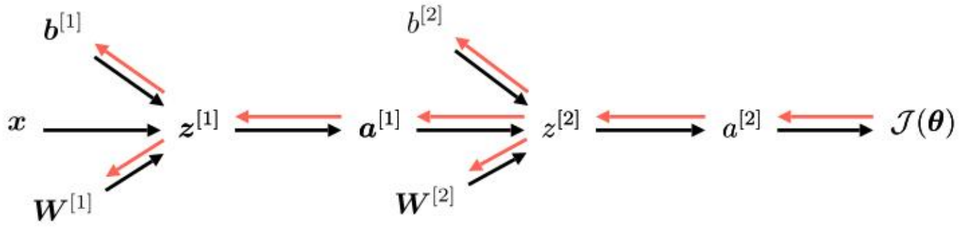
$$\frac{\partial \mathcal{J}}{\partial \boldsymbol{b}^{[1]}} = \frac{\partial (\boldsymbol{z}^{[1]})^{\mathrm{T}}}{\partial \boldsymbol{b}^{[1]}} \cdot \frac{\partial \mathcal{J}}{\partial \boldsymbol{z}^{[1]}} = \frac{\partial \mathcal{J}}{\partial \boldsymbol{z}^{[1]}},$$

$$\frac{\partial \mathcal{J}}{\partial \boldsymbol{W}^{[1]}} = \sum_{k=1}^{d^{[1]}} \frac{\partial z_k^{[1]}}{\partial \boldsymbol{W}^{[1]}} \cdot \frac{\partial \mathcal{J}}{\partial z_k^{[1]}} = \frac{\partial \mathcal{J}}{\partial \boldsymbol{z}^{[1]}} \cdot \boldsymbol{x}^{\mathrm{T}}$$

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

2. It remains to show how to obtain $\partial \mathcal{J}/\partial \boldsymbol{a}^{[1]}$ from $\partial \mathcal{J}/\partial z^{[2]}$

Remarks



1. By the computation graph, we only need to consider ONE more step:

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

Remarks

1. To update parameters in the lth layer, it is enough to know $\partial \mathcal{J} / \partial \boldsymbol{a}^{[l]}$
2. To obtain $\partial \mathcal{J} / \partial \boldsymbol{a}^{[l-1]}$ from $\partial \mathcal{J} / \partial \boldsymbol{z}^{[l]}$, we only need to consider the linear transformation in the lth layer
3. Thus, backpropagation can also be coded up using for-loops

(Batch) Gradient descent algorithm

Step 1. Randomly initialize $\boldsymbol{\theta}^{(0)}$

Step 2. Based on $\boldsymbol{\theta}^{(t)}$ obtain

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

Step 3. Update parameter

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

Step 4. Go back to Step 2 until convergence, or reaching maximum number of iterations

Summary

1. Generalized logistic regression to neural networks
2. Introduced how to obtain gradients for the model parameters
   - Forward propagation: Calculation of output and cost function based on the current model parameters
   - Backpropagation: Obtain gradients by a chain rule
3. Gradient descent algorithm