AI and Deep Learning

2-2 Neural Network with One Hidden Layer II

(Multiple training examples)

Zhonglei Wang

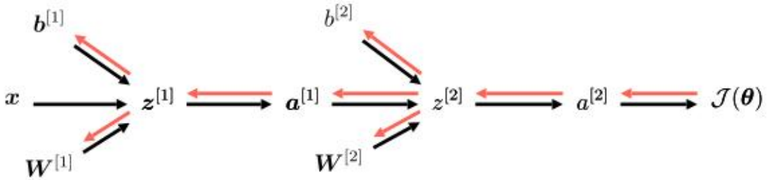
WISE, SOE, CHOW

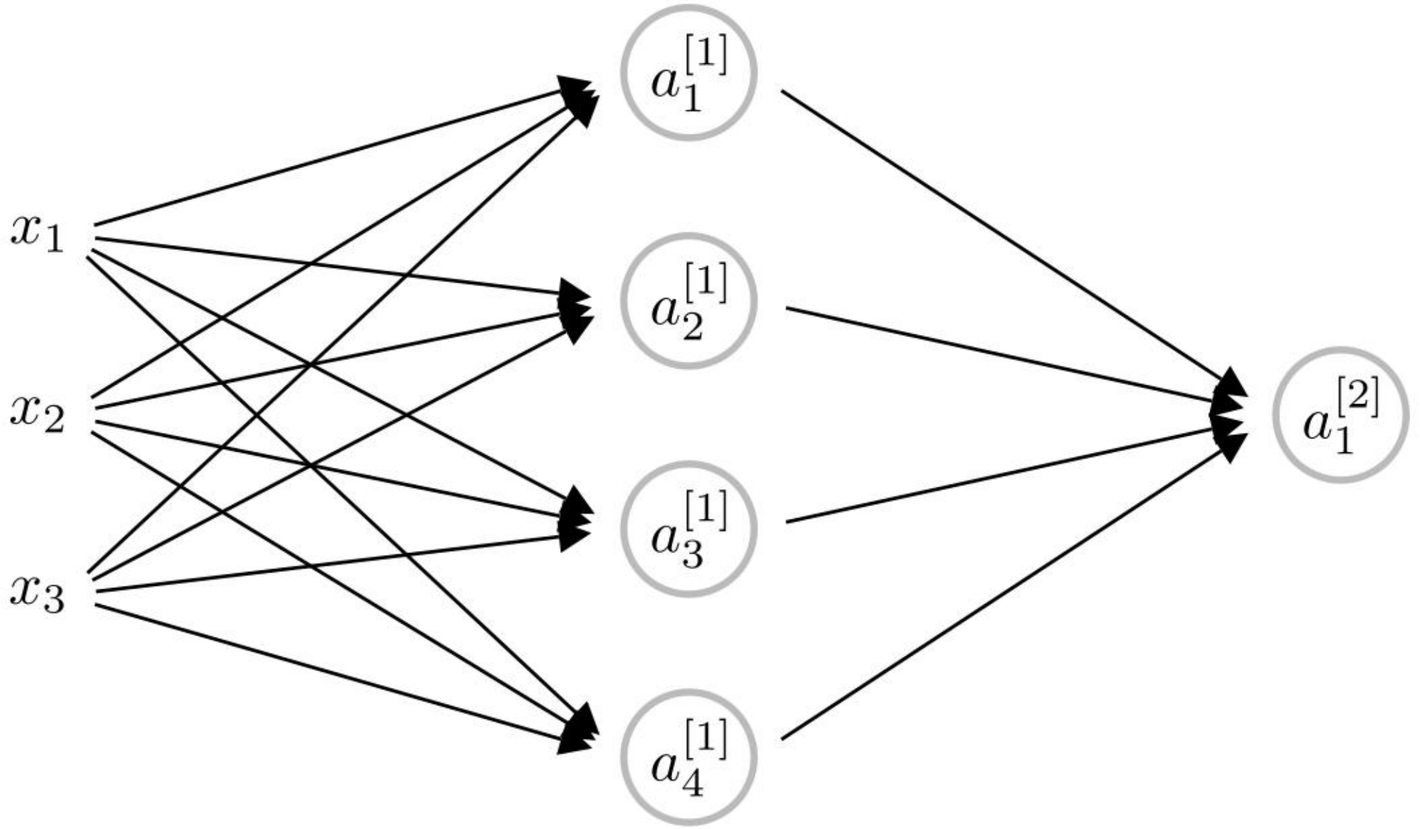
XMU

(Version v1)

Contents

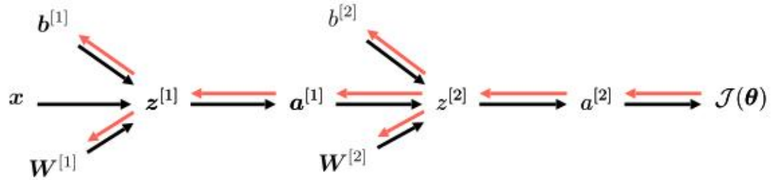
Recall





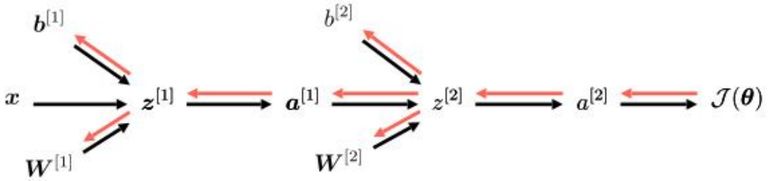
Input layer ($d^{[0]} = 3$) 1st hidden layer ($d^{[1]} = 4$) Output layer ($d^{[2]} = 1$)

Recall



1. Recall that
   - 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. We consider a binary classification problem with $y_i \in \{0, 1\}$

Recall



1. If we have only ONE training example $(\boldsymbol{x}, y)$, forward propagation is

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

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

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

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

   - Broadcasting is used to obtain $\boldsymbol{z}^{[1]}$ and $\boldsymbol{a}^{[1]}$

2. Column vectors are involved above

3. Cost function

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

Forward propagation

1. Suppose we have n training examples $\{(\boldsymbol{x}_i, y_i) : i = 1, \ldots, n\}$
2. Denote $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)^{\mathrm{T}} \in \mathbb{R}^{n \times d}$
3. Forward propagation (using vectorization)

$$\boldsymbol{Z}^{[1]} = (\boldsymbol{b}^{[1]})^{\mathrm{T}} + \boldsymbol{X} \cdot (\boldsymbol{W}^{[1]})^{\mathrm{T}} \in \mathbb{R}^{n \times d^{[1]}},$$

$$\boldsymbol{A}^{[1]} = \sigma(\boldsymbol{Z}^{[1]}) \in \mathbb{R}^{n \times d^{[1]}},$$

$$\boldsymbol{Z}^{[2]} = b^{[2]} + \boldsymbol{A}^{[1]} \cdot (\boldsymbol{W}^{[2]})^{\mathrm{T}} \in \mathbb{R}^{n \times d^{[2]}},$$

$$\boldsymbol{A}^{[2]} = \sigma(\boldsymbol{Z}^{[2]}) \in \mathbb{R}^{n \times d^{[2]}}$$

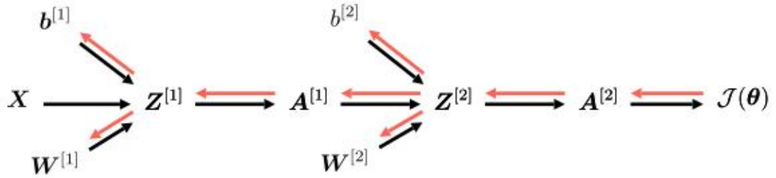
- Broadcasting in Python is implicitly used above

4. Actually, we stack vectors row-wisely

$$\boldsymbol{Z}^{[1]} = (\boldsymbol{b}^{[1]})^{\mathrm{T}} + \begin{pmatrix} \boldsymbol{x}_1^{\mathrm{T}} \\ \boldsymbol{x}_2^{\mathrm{T}} \\ \vdots \\ \boldsymbol{x}_n^{\mathrm{T}} \end{pmatrix} \cdot (\boldsymbol{W}^{[1]})^{\mathrm{T}}$$

$$= \begin{pmatrix} (\boldsymbol{b}^{[1]})^{\mathrm{T}} + \boldsymbol{x}_1^{\mathrm{T}} \cdot (\boldsymbol{W}^{[1]})^{\mathrm{T}} \\ (\boldsymbol{b}^{[1]})^{\mathrm{T}} + \boldsymbol{x}_2^{\mathrm{T}} \cdot (\boldsymbol{W}^{[1]})^{\mathrm{T}} \\ \vdots \\ (\boldsymbol{b}^{[1]})^{\mathrm{T}} + \boldsymbol{x}_n^{\mathrm{T}} \cdot (\boldsymbol{W}^{[1]})^{\mathrm{T}} \end{pmatrix}$$

Forward propagation



1. Suppose we have n training examples $\{(\boldsymbol{x}_i, y_i) : i = 1, \ldots, n\}$

2. Denote $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)^{\mathrm{T}} \in \mathbb{R}^{n \times d}$

3. Forward propagation (using vectorization)

$$\boldsymbol{Z}^{[1]} = (\boldsymbol{b}^{[1]})^{\mathrm{T}} + \boldsymbol{X} \cdot (\boldsymbol{W}^{[1]})^{\mathrm{T}} \in \mathbb{R}^{n \times d^{[1]}},$$

$$\boldsymbol{A}^{[1]} = \sigma(\boldsymbol{Z}^{[1]}) \in \mathbb{R}^{n \times d^{[1]}}, \qquad \boldsymbol{A}^{[1]} = \sigma(\boldsymbol{Z}^{[1]}) = (\sigma(Z_{ij}^{[1]}))$$

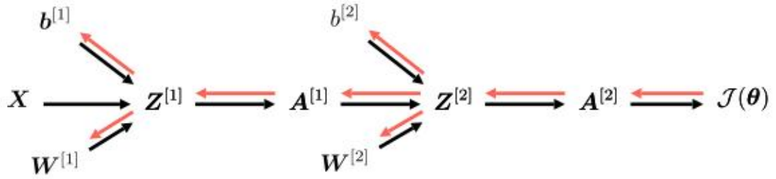
$$\boldsymbol{Z}^{[2]} = b^{[2]} + \boldsymbol{A}^{[1]} \cdot (\boldsymbol{W}^{[2]})^{\mathrm{T}} \in \mathbb{R}^{n \times d^{[2]}}, \qquad (\boldsymbol{Z}^{[1]} = (Z_{ij}^{[1]}); i = 1, \ldots, n; j = 1, \ldots, d^{[1]})$$

$$\boldsymbol{A}^{[2]} = \sigma(\boldsymbol{Z}^{[2]}) \in \mathbb{R}^{n \times d^{[2]}}$$

- Broadcasting in Python is implicitly used above

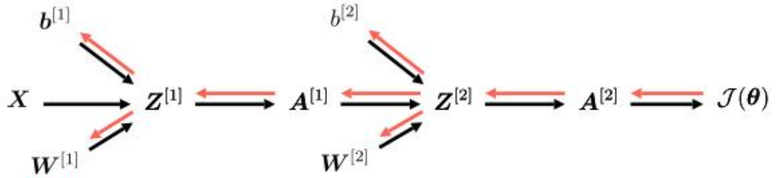
4. Actually, we stack vectors row-wisely

Forward propagation



1. Broadcasting makes Python code simpler, faster, and more memory-efficient
   - Code simplicity and readability
   - Superior memory efficiency: Avoiding temporary copies
   - Massive performance gains: Leveraging optimized C/Fortran code

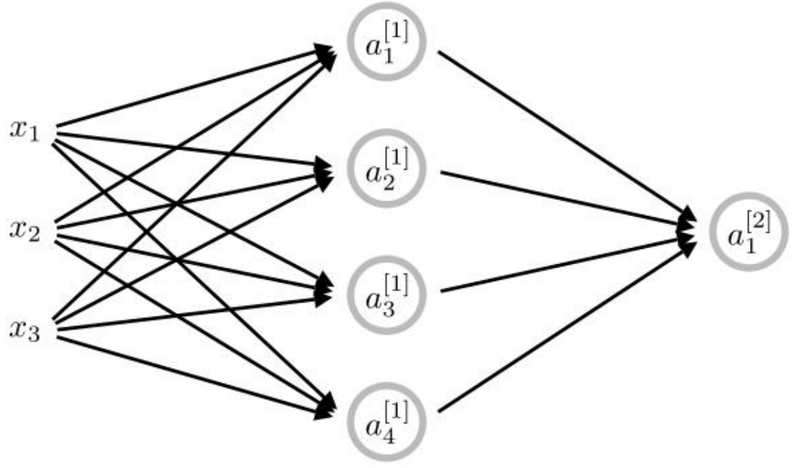
Forward propagation



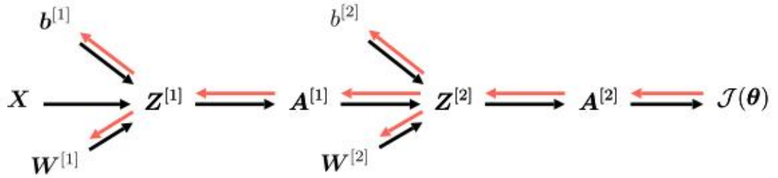
1. For the previous example, we have $d^{[0]} = 3, d^{[1]} = 4, d^{[2]} = 1$

2. Thus,

$$\boldsymbol{Z}^{[1]} = (\boldsymbol{b}^{[1]})^{\mathrm{T}} + \boldsymbol{X} \cdot (\boldsymbol{W}^{[1]})^{\mathrm{T}} \in \mathbb{R}^{n \times 4},$$
$$\boldsymbol{A}^{[1]} = \sigma(\boldsymbol{Z}^{[1]}) \in \mathbb{R}^{n \times 4},$$
$$\boldsymbol{Z}^{[2]} = b^{[2]} + \boldsymbol{A}^{[1]} \cdot (\boldsymbol{W}^{[2]})^{\mathrm{T}} \in \mathbb{R}^{n \times 1},$$
$$\boldsymbol{A}^{[2]} = \sigma(\boldsymbol{Z}^{[2]}) \in \mathbb{R}^{n \times 1}$$



Forward propagation



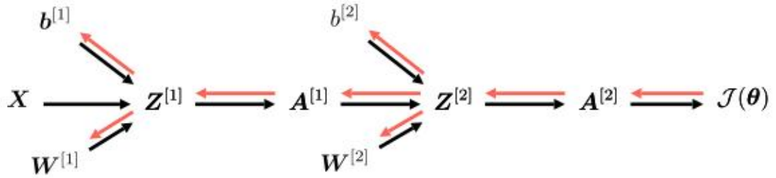
1. Cost function (cross entropy for binary classification problems)

$$\mathcal{J} = -\frac{1}{n}\sum_{i=1}^{n}\left\{y_i \log a_i^{[2]} + (1-y_i)\log\left(1-a_i^{[2]}\right)\right\}$$

- $a_i^{[2]}$: Estimation for $P(y_i \mid \boldsymbol{x}_i)$

2. The form of the cost function is determined by a specific problem

Backpropagation



1. Denote $\mathrm{d}\cdot = \dfrac{\partial \mathcal{J}}{\partial \cdot}$

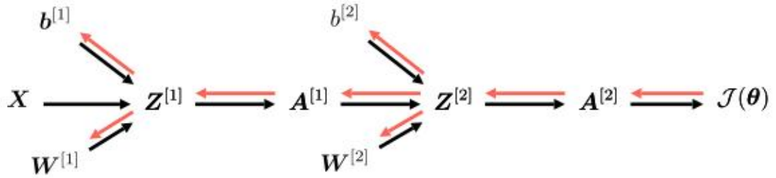
2. Recall: For only one training example$(\boldsymbol{x}, y)$, we have

$$\mathrm{d}z^{[2]} = a^{[2]} - y, \quad \mathrm{d}b^{[2]} = \mathrm{d}z^{[2]}, \quad \mathrm{d}\boldsymbol{W}^{[2]} = \mathrm{d}z^{[2]} \cdot (\boldsymbol{a}^{[1]})^{\mathrm{T}},$$
$$\mathrm{d}\boldsymbol{z}^{[1]} = \mathrm{d}z^{[2]} \cdot \boldsymbol{D} \cdot (\boldsymbol{W}^{[2]})^{\mathrm{T}}, \quad \mathrm{d}\boldsymbol{b}^{[1]} = \mathrm{d}\boldsymbol{z}^{[1]}, \quad \mathrm{d}\boldsymbol{W}^{[1]} = \mathrm{d}\boldsymbol{z}^{[1]} \cdot \boldsymbol{x}^{\mathrm{T}}$$

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

3. Note that
   - $\mathrm{d}z^{[2]}$ can be obtained from $\mathrm{d}a^{[2]}$, determined by the cost function
   - $\mathrm{d}\boldsymbol{z}^{[1]}$ can be obtained from $\mathrm{d}\boldsymbol{a}^{[1]}$, derived from $\mathrm{d}z^{[2]}$ directly by the computation graph

Backpropagation



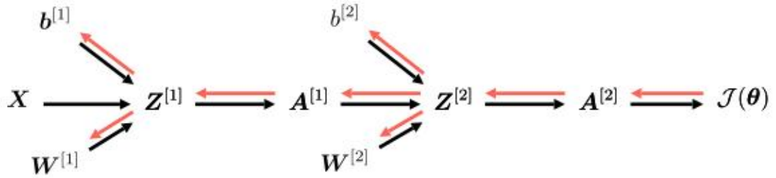
1. For general case, if we have n examples, the cose function is

$$\mathrm{d}\boldsymbol{\theta} = \frac{1}{n}\sum_{i=1}^{n}\frac{\partial \mathcal{L}(y_i, a_i^{[2]})}{\partial \boldsymbol{\theta}}$$

   - $\mathcal{L}(y, a)$: Loss function based on true label y and its estimator a
   - Recall that $\mathcal{L}(y, a) = -\{y \log a + (1-y)\log(1-a)\}$ for binary classification problems

2. Thus, to obtain $\mathrm{d}\boldsymbol{\theta}$, we only need to take average of derivatives for training examples
3. We use vectorization

Backpropagation



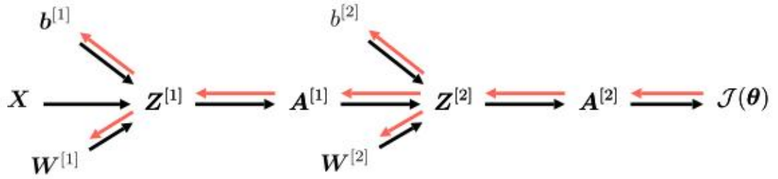
1. Then, we have

$$\mathrm{d}\boldsymbol{Z}^{[2]} = n^{-1}(\boldsymbol{A}^{[2]} - \boldsymbol{Y}), \quad \mathrm{d}b^{[2]} = (\mathrm{d}\boldsymbol{Z}^{[2]})^{\mathrm{T}} \cdot \mathbf{1}, \quad \mathrm{d}\boldsymbol{W}^{[2]} = (\mathrm{d}\boldsymbol{Z}^{[2]})^{\mathrm{T}} \cdot \boldsymbol{A}^{[1]},$$

$$\mathrm{d}\boldsymbol{Z}^{[1]} = (\mathrm{d}\boldsymbol{Z}^{[2]} \cdot \boldsymbol{W}^{[2]}) \circ \sigma'(\boldsymbol{Z}^{[1]}), \quad \mathrm{d}\boldsymbol{b}^{[1]} = (\mathrm{d}\boldsymbol{Z}^{[1]})^{\mathrm{T}} \cdot \mathbf{1}, \quad \mathrm{d}\boldsymbol{W}^{[1]} = (\mathrm{d}\boldsymbol{Z}^{[1]})^{\mathrm{T}} \cdot \boldsymbol{X}$$

- $\sigma'(z)$: Derivative of $\sigma(z)$
  - $\sigma'(z) = \sigma(z)\{1 - \sigma(z)\}$ for sigmoid function $\sigma(z) = \{1 + \exp(-z)\}^{-1}$
- $\sigma'(\boldsymbol{Z}^{[1]}) = (\sigma'(z_{ij}^{[1]})) \in \mathbb{R}^{n \times d^{[1]}}$: Derivatives for each element in $\boldsymbol{Z}^{[1]}$
  - $\boldsymbol{Z}^{[1]} = (z_{ij}^{[1]}) \quad (i = 1, \ldots, n; j = 1, \ldots, d^{[1]})$
- $\circ$: Hadamard production (element-wise production)

Detailed derivation

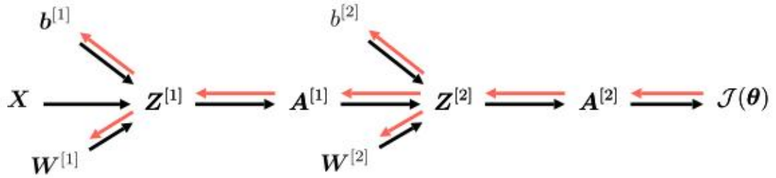


1. We first show the derivation of the derivatives associated with the output layer
2. Recall that the cost function is

$$\mathcal{J} = -\frac{1}{n}\sum_{i=1}^{n}\left\{y_i \log a_i^{[2]} + (1-y_i)\log\left(1-a_i^{[2]}\right)\right\}$$

3. To obtain $\mathrm{d}\boldsymbol{A}^{[2]} = (\partial\mathcal{J}/\partial a_1^{[2]}, \ldots, \partial\mathcal{J}/\partial a_n^{[2]})^{\mathrm{T}} \in \mathbb{R}^{n\times 1}$, we treat $\mathcal{J}$ as a function of $\boldsymbol{A}^{[2]}$
   - $\boldsymbol{A}^{[2]} = (a_1^{[2]}, \ldots, a_n^{[2]})^{\mathrm{T}} \in \mathbb{R}^{n\times 1}$
   - $a_i^{[2]}$ is the estimated probability for $P(y_i = 1 \mid \boldsymbol{x}_i)$
4. We can easily show that $\partial\mathcal{J}/\partial a_i^{[2]} = n^{-1}(a_i^{[2]} - y)/\{a_i^{[2]}(1-a_i^{[2]})\}$

Detailed derivation



1. Thus, we have

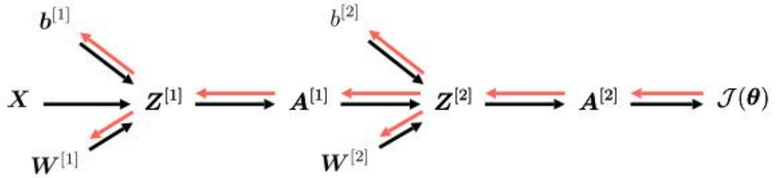
$$\begin{aligned}\mathrm{d}\boldsymbol{A}^{[2]} &= n^{-1}((a_1^{[2]}-y_1)/\{a_1^{[2]}(1-a_1^{[2]})\},\ldots,(a_n^{[2]}-y_n)/\{a_n^{[2]}(1-a_n^{[2]})\})^{\mathrm{T}}\\ &= n^{-1}\boldsymbol{D}^{-1}\cdot(\boldsymbol{A}^{[2]}-\boldsymbol{Y})\end{aligned}$$

   - $\boldsymbol{D}\in\mathbb{R}^{n\times n}$ is a diagonal matrix, whose (i,i)th entry is $a_i^{[2]}(1-a_i^{[2]})$

2. To obtain $\mathrm{d}\boldsymbol{Z}^{[2]}=(\partial\mathcal{J}/\partial z_1^{[2]},\ldots,\partial\mathcal{J}/\partial z_n^{[2]})^{\mathrm{T}}\in\mathbb{R}^{n\times 1}$, we use the chain rule and treat $\mathcal{J}$ as a function of $\boldsymbol{A}^{[2]}$, and $\boldsymbol{A}^{[2]}$ as a function of $\boldsymbol{Z}^{[2]}$

   - $\boldsymbol{Z}^{[2]}=(z_1^{[2]},\ldots,z_n^{[2]})^{\mathrm{T}}\in\mathbb{R}^{n\times 1}$
   - $z_i^{[2]}$ is the result after linear transformation in the second layer for the ith training example

Detailed derivation



1. Since both $\boldsymbol{A}^{[2]}$ and $\boldsymbol{Z}^{[2]}$ are column vectors, we have

$$\mathrm{d}\boldsymbol{Z}^{[2]} = \frac{\partial \mathcal{J}}{\partial \boldsymbol{Z}^{[2]}} = \frac{\partial (\boldsymbol{A}^{[2]})^{\mathrm{T}}}{\partial \boldsymbol{Z}^{[2]}} \cdot \frac{\partial \mathcal{J}}{\partial \boldsymbol{A}^{[2]}} = \frac{\partial (\boldsymbol{A}^{[2]})^{\mathrm{T}}}{\partial \boldsymbol{Z}^{[2]}} \cdot n^{-1} \cdot \boldsymbol{D}^{-1} \cdot (\boldsymbol{A}^{[2]} - \boldsymbol{Y})$$

2. Notice that

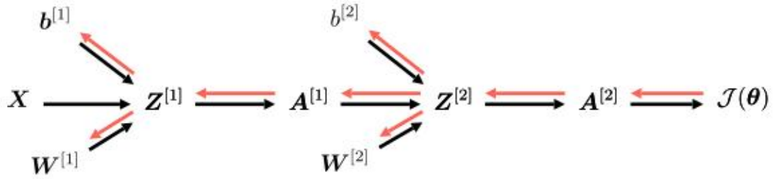
$$\frac{\partial (\boldsymbol{A}^{[2]})^{\mathrm{T}}}{\partial \boldsymbol{Z}^{[2]}} = \boldsymbol{D}$$

   - Since $a_i^{[2]}$ is only a function of $z_i^{[2]}$, $\partial (\boldsymbol{A}^{[2]})^{\mathrm{T}} / \partial \boldsymbol{Z}^{[2]}$ is diagonal
   - Since $a_i^{[2]} = \sigma(z_i^{[2]})$, $\partial a_i^{[2]} / \partial z_i^{[2]} = a_i^{[2]}(1 - a_i^{[2]})$, exactly the (i,i)th entry of $\boldsymbol{D}$

3. Thus, we have

$$\mathrm{d}\boldsymbol{Z}^{[2]} = n^{-1}(\boldsymbol{A}^{[2]} - \boldsymbol{Y})$$

Detailed derivation

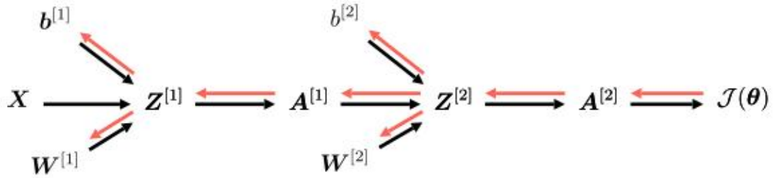


1. To obtain $\mathrm{d}b^{[2]} = \partial\mathcal{J}/\partial b^{[2]}$, we use the chain rule and treat $\mathcal{J}$ as a function of $\boldsymbol{Z}^{[2]}$, and $\boldsymbol{Z}^{[2]}$ as a function of $b^{[2]}$
2. Thus, we have

$$\mathrm{d}b^{[2]} = \frac{\partial\mathcal{J}}{\partial b^{[2]}} = \frac{\partial(\boldsymbol{Z}^{[2]})^{\mathrm{T}}}{\partial b^{[2]}} \cdot \frac{\partial\mathcal{J}}{\partial \boldsymbol{Z}^{[2]}} = \mathbf{1}^{\mathrm{T}} \cdot \mathrm{d}\boldsymbol{Z}^{[2]} = (\mathrm{d}\boldsymbol{Z}^{[2]})^{\mathrm{T}} \cdot \mathbf{1}$$

- $\mathbf{1} = (1, \ldots, 1)^{\mathrm{T}}$ is a column vector of length n with all elements being 1
- Each element in $\boldsymbol{Z}^{[2]}$ is a function of $b^{[2]}$: $z_i^{[2]} = b^{[2]} + \boldsymbol{W}^{[2]} \cdot \boldsymbol{a}_i^{[1]}$
- Thus, $\partial z_i^{[2]}/\partial b^{[2]} = 1$ and $\partial \boldsymbol{Z}^{[2]}/\partial b^{[2]} = (\partial z_1^{[2]}/\partial b^{[2]}, \ldots, \partial z_n^{[2]}/\partial b^{[2]})^{\mathrm{T}} = \mathbf{1}$

Detailed derivation

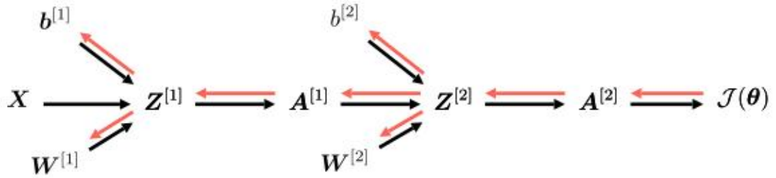


1. To obtain $\mathrm{d}\boldsymbol{W}^{[2]} = \partial\mathcal{J}/\partial\boldsymbol{W}^{[2]}$, we use the chain rule and treat $\mathcal{J}$ as a function of $\boldsymbol{Z}^{[2]}$, and $\boldsymbol{Z}^{[2]}$ as a function of $\boldsymbol{W}^{[2]}$
2. Since $\boldsymbol{W}^{[2]}$ is a row vector, not a column one, we have

$$\mathrm{d}\boldsymbol{W}^{[2]} = \sum_{i=1}^{n} \frac{\partial z_i^{[2]}}{\partial \boldsymbol{W}^{[2]}} \cdot \frac{\partial \mathcal{J}}{\partial z_i^{[2]}} = \left(\frac{\partial \mathcal{J}}{\partial z_1^{[2]}}, \ldots, \frac{\partial \mathcal{J}}{\partial z_n^{[2]}} \right) \cdot \begin{pmatrix} (\boldsymbol{a}_1^{[1]})^{\mathrm{T}} \\ \vdots \\ (\boldsymbol{a}_n^{[1]})^{\mathrm{T}} \end{pmatrix} = (\mathrm{d}\boldsymbol{Z}^{[2]})^{\mathrm{T}} \cdot \boldsymbol{A}^{[1]}$$

- Each element in $\boldsymbol{Z}^{[2]}$ is a function of $\boldsymbol{W}^{[2]}$: $z_i^{[2]} = b^{[2]} + \boldsymbol{W}^{[2]} \cdot \boldsymbol{a}_i^{[1]}$
- Thus, $\partial z_i^{[2]}/\partial \boldsymbol{W}^{[2]} = (\boldsymbol{a}_i^{[1]})^{\mathrm{T}}$
- The last equation holds by vectorization

Detailed derivation

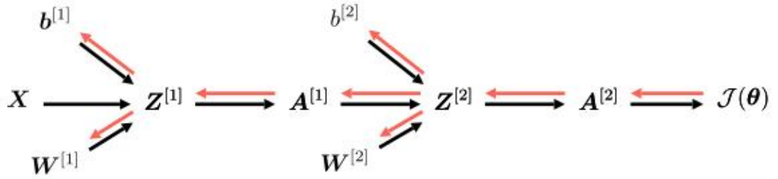


1. Since $(\boldsymbol{W}^{[2]})^{\mathrm{T}}$ is a column vector, another way to obtain $\mathrm{d}\boldsymbol{W}^{[2]}$ is

$$\mathrm{d}\boldsymbol{W}^{[2]} = \left(\frac{\partial \mathcal{J}}{\partial (\boldsymbol{W}^{[2]})^{\mathrm{T}}}\right)^{\mathrm{T}} = \left(\frac{\partial (\boldsymbol{Z}^{[2]})^{\mathrm{T}}}{\partial (\boldsymbol{W}^{[2]})^{\mathrm{T}}} \cdot \frac{\partial \mathcal{J}}{\partial \boldsymbol{Z}^{[2]}}\right)^{\mathrm{T}} = (\mathrm{d}\boldsymbol{Z}^{[2]})^{\mathrm{T}} \cdot \boldsymbol{A}^{[1]}$$

- Each element in $\boldsymbol{Z}^{[2]}$ is a function of $\boldsymbol{W}^{[2]}$: $z_i^{[2]} = b^{[2]} + \boldsymbol{W}^{[2]} \cdot \boldsymbol{a}_i^{[1]}$
- Thus, $\partial z_i^{[2]} / \partial (\boldsymbol{W}^{[2]})^{\mathrm{T}} = \boldsymbol{a}_i^{[1]}$ and $\partial (\boldsymbol{Z}^{[2]})^{\mathrm{T}} / \partial (\boldsymbol{W}^{[2]})^{\mathrm{T}} = (\boldsymbol{a}_1^{[1]}, \ldots, \boldsymbol{a}_n^{[1]}) = (\boldsymbol{A}^{[1]})^{\mathrm{T}}$

Detailed derivation



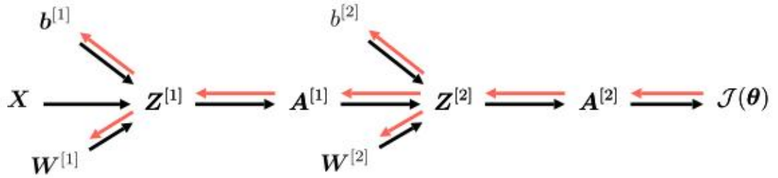
1. To obtain $\mathrm{d}\boldsymbol{A}^{[1]} = \partial\mathcal{J}/\partial\boldsymbol{A}^{[1]}$, we use the chain rule and treat $\mathcal{J}$ as a function of $\boldsymbol{Z}^{[2]}$, and $\boldsymbol{Z}^{[2]}$ as a function of $\boldsymbol{A}^{[1]}$

2. Since $\boldsymbol{A}^{[1]} \in \mathbb{R}^{n\times d^{[1]}}$ is a matrix, not a column one, we have

$$\mathrm{d}\boldsymbol{A}^{[1]} = \sum_{i=1}^{n} \frac{\partial z_i^{[2]}}{\partial \boldsymbol{A}^{[1]}} \cdot \frac{\partial \mathcal{J}}{\partial z_i^{[2]}} = \sum_{i=1}^{n} \frac{\partial \mathcal{J}}{\partial z_i^{[2]}} \cdot \boldsymbol{e}_i \cdot \boldsymbol{W}^{[2]} = \frac{\partial \mathcal{J}}{\partial \boldsymbol{Z}^{[2]}} \cdot \boldsymbol{W}^{[2]} = \mathrm{d}\boldsymbol{Z}^{[2]} \cdot \boldsymbol{W}^{[2]}$$

- $\partial z_i^{[2]}/\partial \boldsymbol{A}^{[1]} \in \mathbb{R}^{n\times d^{[1]}}$ has the same dimension with $\boldsymbol{A}^{[1]}$
- $z_i^{[2]}$ is a only function of the ith row of $\boldsymbol{A}^{[1]}$: $z_i^{[2]} = b^{[2]} + \boldsymbol{W}^{[2]} \cdot \boldsymbol{a}_i^{[1]}$
- Thus, $\partial z_i^{[2]}/\partial (\boldsymbol{a}_i^{[1]})^{\mathrm{T}} = \boldsymbol{W}^{[2]}$ and $\partial z_i^{[2]}/\partial \boldsymbol{A}^{[1]} = \boldsymbol{e}_i \cdot \boldsymbol{W}^{[2]}$
- $\boldsymbol{e}_i = (0, \ldots, 1, \ldots, 0)^{\mathrm{T}}$ is a column vector of length n; all elements are zero but the ith being 1

Detailed derivation



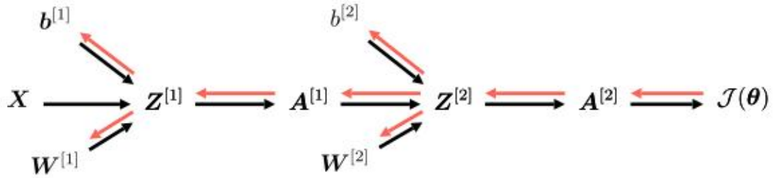
1. To get $\mathrm{d}\boldsymbol{Z}^{[1]} = \partial \mathcal{J} / \partial \boldsymbol{Z}^{[1]}$, we treat $\mathcal{J}$ as a function of $\boldsymbol{A}^{[1]}$, and $\boldsymbol{A}^{[1]}$ as a function of $\boldsymbol{Z}^{[1]}$

2. Since $\boldsymbol{A}^{[1]} \in \mathbb{R}^{n \times d^{[1]}}$, we have

$$\mathrm{d}\boldsymbol{Z}^{[1]} = \sum_{i=1}^{n} \sum_{j=1}^{d^{[1]}} \frac{\partial a_{ij}^{[1]}}{\partial \boldsymbol{Z}^{[1]}} \cdot \frac{\partial \mathcal{J}}{\partial a_{ij}^{[1]}} = \sum_{i=1}^{n} \sum_{j=1}^{d^{[1]}} \frac{\partial \mathcal{J}}{\partial a_{ij}^{[1]}} \cdot \sigma'(z_{ij}^{[1]}) \cdot \boldsymbol{e}_i \cdot \boldsymbol{h}_j^{\mathrm{T}} = \mathrm{d}\boldsymbol{A}^{[1]} \circ \sigma'(\boldsymbol{Z}^{[1]})$$

- $\partial a_{ij}^{[1]} / \partial \boldsymbol{Z}^{[1]} \in \mathbb{R}^{n \times d^{[1]}}$ has the same dimension with $\boldsymbol{Z}^{[1]}$
- $a_{ij}^{[1]}$ is a only function of the (i, j)th element of $\boldsymbol{Z}^{[1]}$: $a_{ij}^{[1]} = \sigma(z_{ij}^{[1]})$
- Thus, $\partial a_{ij}^{[1]} / \partial z_{ij}^{[1]} = \sigma'(z_{ij}^{[1]})$ and $\partial a_{ij}^{[1]} / \partial \boldsymbol{Z}^{[1]} = \sigma'(z_{ij}^{[1]}) \cdot \boldsymbol{e}_i \cdot \boldsymbol{h}_j^{\mathrm{T}}$
- $\boldsymbol{e}_i = (0, \ldots, 1, \ldots, 0)^{\mathrm{T}}$ is a column vector of length n; all elements are zero but the ith being 1
- $\boldsymbol{h}_j = (0, \ldots, 1, \ldots, 0)^{\mathrm{T}}$ is a column vector of length $d^{[1]}$; all are zero but the jth being 1

Detailed derivation

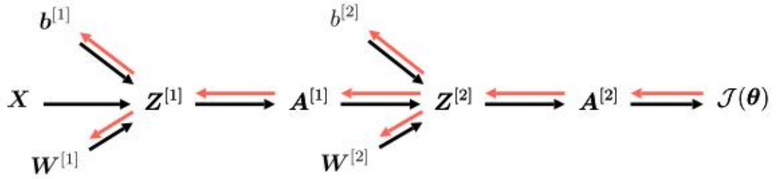


1. To obtain $\mathrm{d}\boldsymbol{b}^{[1]} = \partial\mathcal{J}/\partial\boldsymbol{b}^{[1]}$, we treat $\mathcal{J}$ as a function of $\boldsymbol{Z}^{[1]} = (\boldsymbol{z}_1^{[1]}, \ldots, \boldsymbol{z}_n^{[1]})^{\mathrm{T}}$, and $\boldsymbol{Z}^{[1]}$ as a function of $\boldsymbol{b}^{[1]}$
2. Since $\boldsymbol{b}^{[1]} \in \mathbb{R}^{d^{[1]}\times 1}$ is a column vector, we have

$$\mathrm{d}\boldsymbol{b}^{[1]} = \sum_{i=1}^{n} \frac{\partial(\boldsymbol{z}_i^{[1]})^{\mathrm{T}}}{\partial\boldsymbol{b}^{[1]}} \cdot \frac{\partial\mathcal{J}}{\partial\boldsymbol{z}_i^{[1]}} = \sum_{i=1}^{n} \boldsymbol{I} \cdot \frac{\partial\mathcal{J}}{\partial\boldsymbol{z}_i^{[1]}} = \sum_{i=1}^{n} \frac{\partial\mathcal{J}}{\partial\boldsymbol{z}_i^{[1]}} = (\mathrm{d}\boldsymbol{Z}^{[1]})^{\mathrm{T}} \cdot \boldsymbol{1}$$

- Each row of $\boldsymbol{Z}^{[1]}$ is a function of $\boldsymbol{b}^{[1]}$: $(\boldsymbol{z}_i^{[1]})^{\mathrm{T}} = (\boldsymbol{b}^{[1]})^{\mathrm{T}} + \boldsymbol{x}_i^{\mathrm{T}} \cdot (\boldsymbol{W}^{[1]})^{\mathrm{T}}$
- Thus, $\partial(\boldsymbol{z}_i^{[1]})^{\mathrm{T}}/\partial\boldsymbol{b}^{[1]} = \boldsymbol{I}$
- $\boldsymbol{I}$ is an identity matrix of dimension $d^{[1]} \times d^{[1]}$

Detailed derivation



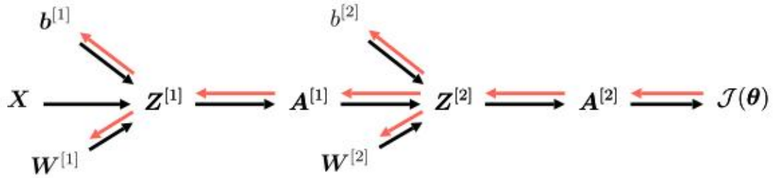
1. To obtain $\mathrm{d}\boldsymbol{W}^{[1]} = \partial\mathcal{J}/\partial\boldsymbol{W}^{[1]}$, we treat $\mathcal{J}$ as a function of $\boldsymbol{Z}^{[1]} = (z_{ij}^{[1]})$, and $\boldsymbol{Z}^{[1]}$ as a function of $\boldsymbol{W}^{[1]}$

2. Since $\boldsymbol{W}^{[1]} \in \mathbb{R}^{d^{[1]} \times d^{[0]}}$ is a matrix, we have

$$\mathrm{d}\boldsymbol{W}^{[1]} = \sum_{i=1}^{n}\sum_{j=1}^{d^{[1]}} \frac{\partial \boldsymbol{z}_{ij}^{[1]}}{\partial \boldsymbol{W}^{[1]}} \cdot \frac{\partial \mathcal{J}}{\partial \boldsymbol{z}_{ij}^{[1]}} = \sum_{i=1}^{n}\sum_{j=1}^{d^{[1]}} \frac{\partial \mathcal{J}}{\partial \boldsymbol{z}_{ij}^{[1]}} \cdot \boldsymbol{h}_j \cdot \boldsymbol{x}_i^{\mathrm{T}} = \sum_{i=1}^{n} \frac{\partial \mathcal{J}}{\partial \boldsymbol{z}_{i}^{[1]}} \cdot \boldsymbol{x}_i^{\mathrm{T}} = (\mathrm{d}\boldsymbol{Z}^{[1]})^{\mathrm{T}} \cdot \boldsymbol{X}$$

- $z_{ij}^{[1]}$ is only a function of the jth row of $\boldsymbol{W}^{[1]} = (\boldsymbol{w}_1^{[1]}, \ldots, \boldsymbol{w}_{d^{[1]}}^{[1]})^{\mathrm{T}}$: $z_{ij}^{[1]} = b_j^{[1]} + (\boldsymbol{w}_j^{[1]})^{\mathrm{T}} \cdot \boldsymbol{x}_i$
- Thus, $\partial z_{ij}^{[1]}/\partial \boldsymbol{W}^{[1]} = \boldsymbol{h}_j \cdot \boldsymbol{x}_i^{\mathrm{T}}$
- $\boldsymbol{h}_j = (0, \ldots, 1, \ldots, 0)^{\mathrm{T}}$ is a column vector of length $d^{[1]}$; all are zero but the jth being 1

Remarks



1. For convenience, let $\boldsymbol{A}^{[0]} = \boldsymbol{X}$
2. Forward propagation and backpropagation can be coded up using for-loops
   - Forward propagation: Based on the current model parameters, $\boldsymbol{A}^{[l-1]}$ is the "input' for $\boldsymbol{Z}^{[l]}$ and $\boldsymbol{A}^{[l]}$
   - Backpropagation: $\mathrm{d}\boldsymbol{A}^{[l]}$ is the "input" to obtain $\mathrm{d}bZ^{[l]}$, $\mathrm{d}\boldsymbol{b}^{[l]}$, $\mathrm{d}\boldsymbol{W}^{[l]}$, $\mathrm{d}\boldsymbol{Z}^{[l-1]}$ and $\mathrm{d}\boldsymbol{A}^{[l-1]}$

(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. Extend the single-hidden-layer neural network from single-sample computation to vectorized training with multiple training samples.
   - Data: Stack features row-wisely to obtain a design matrix $\boldsymbol{X}$
   - Forward propagation: Use current parameters to obtain cost, and broadcasting is applied to improve computation efficiency
   - Backpropagation: Obtain average gradients by a chain rule and vectorization
   - Gradient descent algorithm