13. Matrix algebra and Backpropagation

M.A.Z. Chowdhury and M.A. Oehlschlaeger

Department of Mechanical, Aerospace and Nuclear Engineering
Rensselaer Polytechnic Institute
Troy, New York

chowdm@rpi.edu
oehlsm@rpi.edu

MANE 4962 and 6962

Regular announcement

- Quiz 4 Today.
- Quiz 4 is based on notes from Lecture 10.
- HW 4 is due March 2.
- Initial project proposal due March 2.

Outline

- HW4, Linear Algebra, and backpropagation
- Further discussion about the initial and revised project proposal submission
- We solved regression problems using a fully connected neural network, $y \in \mathbb{R}$.
  outputs one number, we used mse.
- We solved a binary classification problem using a fully connected neural network $y \in [0, 1]$.
  outputs one number, we used binary crossentropy.
- Solve a multi-class classification problem using a fully connected neural networks $y \in \mathbb{R}^{K}$, K is the number of classes.
  outputs K numbers, we used categorical crossentropy.
- For project, We can solve regression problems using a fully connected neural network, $y \in \mathbb{R}^{u}$.
  outputs u numbers. Use regression cost function, i.e., mse-like.

Representing Vectors and Matrices

Vector with n elements, $x \in \mathbb{R}^n$

$(x_1, x_2 \ldots x_n)$

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

Matrix with m rows & n columns, such that elements of it are real numbers, $A \in \mathbb{R}^{m \times n}$

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} = \begin{bmatrix} | & | & & | \\ a^1 & a^2 & \cdots & a^n \\ | & | & & | \end{bmatrix} = \begin{bmatrix} — & a_1^T & — \\ — & a_2^T & — \\ & \vdots & \\ — & a_m^T & — \end{bmatrix}$$

Identity & Diagonal Matrices

Identity matrix, $I \in \mathbb{R}^{n \times n}$, is a square matrix and zeros everywhere else.

$$I_{ij} = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases}$$

Special Property For all $A \in \mathbb{R}^{m \times n}$

$$AI = A = IA$$

For a diagonal matrix, $D = \text{diag}(d_1, d_2, \ldots, d_n)$

$$D_{ij} = \begin{cases} d_i, & i = j \\ 0, & i \neq j \end{cases}$$

So, $I = \text{diag}(1, 1, \ldots, 1)$

Vector - vector product

① $x^T y \in \mathbb{R} = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \sum x_i y_i$ Scalar

② $xy^T \in \mathbb{R}^{m \times n} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \begin{bmatrix} y_1 & y_2 & \cdots & y_n \end{bmatrix} = \begin{bmatrix} x_1 y_1 & x_1 y_2 & \cdots & x_1 y_n \\ x_2 y_1 & x_2 y_2 & \cdots & x_2 y_n \\ \vdots & \vdots & \ddots & \vdots \\ x_m y_1 & x_m y_2 & \cdots & x_m y_n \end{bmatrix}$ Matrix

① is called vector-vector inner product, it is also known as dot product.

② is called vector-vector dot product.

Matrix - Vector Product

(1) $$y = Ax = \begin{bmatrix} - & a_1^T & - \\ - & a_2^T & - \\ & \vdots & \\ - & a_m^T & - \end{bmatrix} x = \begin{bmatrix} a_1^T x \\ a_2^T x \\ \vdots \\ a_m^T x \end{bmatrix}$$

in (1) we multiplied by rows

$$y = Ax = \begin{bmatrix} | & | & & | \\ a^1 & a^2 & \cdots & a^n \\ | & | & & | \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} | \\ a^1 \\ | \end{bmatrix} x_1 + \begin{bmatrix} | \\ a^2 \\ | \end{bmatrix} x_2 \cdots + \begin{bmatrix} | \\ a^n \\ | \end{bmatrix} x_n$$

linear combination

in (2) we multiplied by columns

y is a linear combination of columns of A

Matrix-Vector product (contd.)

we can multiply on the left by a row vector

$$y^T = x^T A = x^T \begin{bmatrix} | & | & & | \\ a^1 & a^2 & \cdots & a^n \\ | & | & & | \end{bmatrix} = \begin{bmatrix} x^T a^1 & x^T a^2 & \cdots & x^T a^n \end{bmatrix}$$

or

$$y^T = \begin{bmatrix} x_1 & x_2 & \cdots & x_m \end{bmatrix} \begin{bmatrix} - & a_1^T & - \\ - & a_2^T & - \\ & \vdots & \\ - & a_m^T & - \end{bmatrix} = x_1 \begin{bmatrix} - & a_1^T & - \end{bmatrix} + x_2 \begin{bmatrix} - & a_2^T & - \end{bmatrix} + \cdots + x_m \begin{bmatrix} - & a_m^T & - \end{bmatrix}$$

y^T is a linear combination of rows of A

Matrix-matrix multiplication

1. Using the concept of vector-vector inner (dot) product

$$C = AB = \begin{bmatrix} - & a_1^T & - \\ - & a_2^T & - \\ & \vdots & \\ - & a_m^T & - \end{bmatrix} = \begin{bmatrix} | & | & & | \\ b^1 & b^2 & \cdots & b^n \\ | & | & & | \end{bmatrix}$$

$$= \begin{bmatrix} a_1^T b^1 & a_1^T b^2 & \cdots & a_1^T b^n \\ a_2^T b^1 & a_2^T b^2 & \cdots & a_2^T b^n \\ \vdots & \vdots & \ddots & \vdots \\ a_m^T b^1 & a_m^T b^2 & \cdots & a_m^T b^n \end{bmatrix}$$

Matrix-matrix multiplication (contd.)

2. Using the concept of outer product

$$C = AB = \begin{bmatrix} | & | & & | \\ a^1 & a^2 & \cdots & a^n \\ | & | & & | \end{bmatrix} \begin{bmatrix} - & b_1^T & - \\ - & b_2^T & - \\ & \vdots & \\ - & b_n^T & - \end{bmatrix} = \sum_{i=1}^{n} a^i b_i^T$$

matrix-matrix multiplication (contd.)

3. As a set of matrix-vector products

$$C = AB = A\begin{bmatrix} | & | & & | \\ b^1 & b^2 & \cdots & b^n \\ | & | & & | \end{bmatrix} = \begin{bmatrix} | & | & & | \\ Ab^1 & Ab^2 & \cdots & Ab^n \\ | & | & & | \end{bmatrix}$$

The i-th column of C is given by the matrix-vector product with the vector on the right, $c_i = Ab_i$.

$\times$ Einstein Notation

Column

Matrix-matrix multiplication (contd.)

4. As a set of vector-matrix products

$$C = AB = \begin{bmatrix} - a_1^T - \\ - a_2^T - \\ \vdots \\ - a_m^T - \end{bmatrix} B = \begin{bmatrix} - a_1^T B - \\ - a_2^T B - \\ \vdots \\ - a_m^T B - \end{bmatrix}$$

Properties of matrix-matrix multiplication

① Associative : $(AB)C = A(BC)$

② Distributive : $A(B+C) = AB + AC$

③ Not necessarily commutative

For example, if $A \in \mathbb{R}^{m \times n}$, $B \in \mathbb{R}^{n \times a}$ and $m \neq n \neq a$, then BA does not exist.

$$AB \neq BA$$

Hadamard or Elementwise Product

→ Matrix Product, $C = AB$

$$\Rightarrow C_{ij} = \sum_{k=1}^{n} A_{ik} B_{kj}$$

→ Hadamard Product, $C = A \odot B$

↳ A, B, C are of the same size

↳ multiply elements in A and B at same position. $(A \odot B)_{ij} = A_{ij} B_{ij}$

↳ Example

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \odot \begin{bmatrix} 1 & 5 & 6 \\ 1 & 7 & 8 \end{bmatrix} = \begin{bmatrix} 1 & 10 & 18 \\ 4 & 35 & 48 \end{bmatrix}$$

Transpose

The transpose operation flips the rows and columns. Given, $A \in \mathbb{R}^{m \times n}$, its transpose $A^T \in \mathbb{R}^{n \times m}$ whose entries are given by,

$$(A^T)_{ij} = A_{ji}$$

Properties

$$(A^T)^T = A$$

$$(AB)^T = B^T A^T$$

$$(A+B)^T = A^T + B^T$$

if $A = A^T$, then A is symmetric.

Trace

Trace of a square matrix $A \in \mathbb{R}^{n \times n}$ is denoted by $\text{tr} A$ or $\text{tr}(A)$ is the sum of diagonal elements of A.

$$\text{tr} A = \sum_{i=1}^{n} A_{ii}$$ ← Einstein's

Properties : Consider $A \in \mathbb{R}^{n \times n}$ & $B \in \mathbb{R}^{n \times n}$

1. $\text{tr} A = \text{tr} A^T$
2. $\text{tr}(A+B) = \text{tr} A + \text{tr} B$
3. $\text{tr}(\alpha A) = \alpha \, \text{tr} A, \ \alpha \in \mathbb{R}$
4. $\text{tr}(AB) = \text{tr}(BA)$, for A, B such that AB is square
5. $\text{tr}(ABC) = \text{tr}(BCA) = \text{tr}(CAB)$, such that ABC is square. Can be expanded more

Rank of a matrix

Column rank of $A \in \mathbb{R}^{m \times n}$ is the largest number of columns of A that constitute a linearly independent set.

Row rank of $A \in \mathbb{R}^{m \times n}$ is the largest number of rows of A that constitute a linearly independent set.

For $A \in \mathbb{R}^{m \times n}$, $\boxed{\text{column rank}}$ and $\boxed{\text{row rank}}$ are equal, so generally rank is represented by rank(A).

Properties of the rank

- For $A \in \mathbb{R}^{m \times n}$, $\text{rank}(A) \leq \min(m, n)$ → rows → columns
- If $\text{rank}(A) = \min(m, n)$, then A is $\boxed{\text{full rank.}}$
- For $A \in \mathbb{R}^{m \times n}$, $\text{rank}(A) = \text{rank}(A^T)$
- For $A \in \mathbb{R}^{m \times p}$, $B \in \mathbb{R}^{p \times n}$, $\text{rank}(AB) \leq \min(\text{rank}(A), \text{rank}(B))$
- For $A, B \in \mathbb{R}^{m \times n}$, $\text{rank}(A+B) \leq \text{rank}(A) + \text{rank}(B)$

Inverse of A (A is a square matrix)

→ inverse of a square matrix $A \in \mathbb{R}^{n \times n}$ is denoted by A^{-1}, and is the unique matrix such that $A^{-1}A = I = AA^{-1}$

→ A is invertible or non-singular if A^{-1} exists and non-invertible or singular otherwise

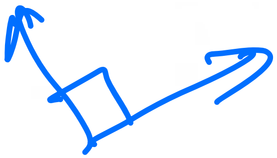
→ For square matrix A to have an inverse A^{-1}, A must be full rank.

Properties of inverse of A (A is a square matrix)

$$(A^{-1})^{-1} = A$$

$$(AB)^{-1} = B^{-1} A^{-1}$$

$$(A^{-1})^T = (A^T)^{-1}$$, sometimes denoted A^{-T}



Orthogonal Matrices

→ Two vectors are orthogonal, if $\underline{x^T y} = 0$

→ A vector is normalized if $\|x\|_2 = 1$

→ A square matrix, $A \in \mathbb{R}^{n \times n}$ is orthogonal if all its columns are orthogonal to each other and normalized.

→ Such columns are called $\boxed{\text{orthonormal}}$

Properties of orthogonal matrices

→ The inverse of an orthogonal matrix is its transpose.

$$A^{-1} = A^T \quad (A \text{ is orthogonal})$$

$$A^T A = I = A A^T$$

→ Operating on a vector with an orthogonal matrix does not change its Euclidean norm,

scikit

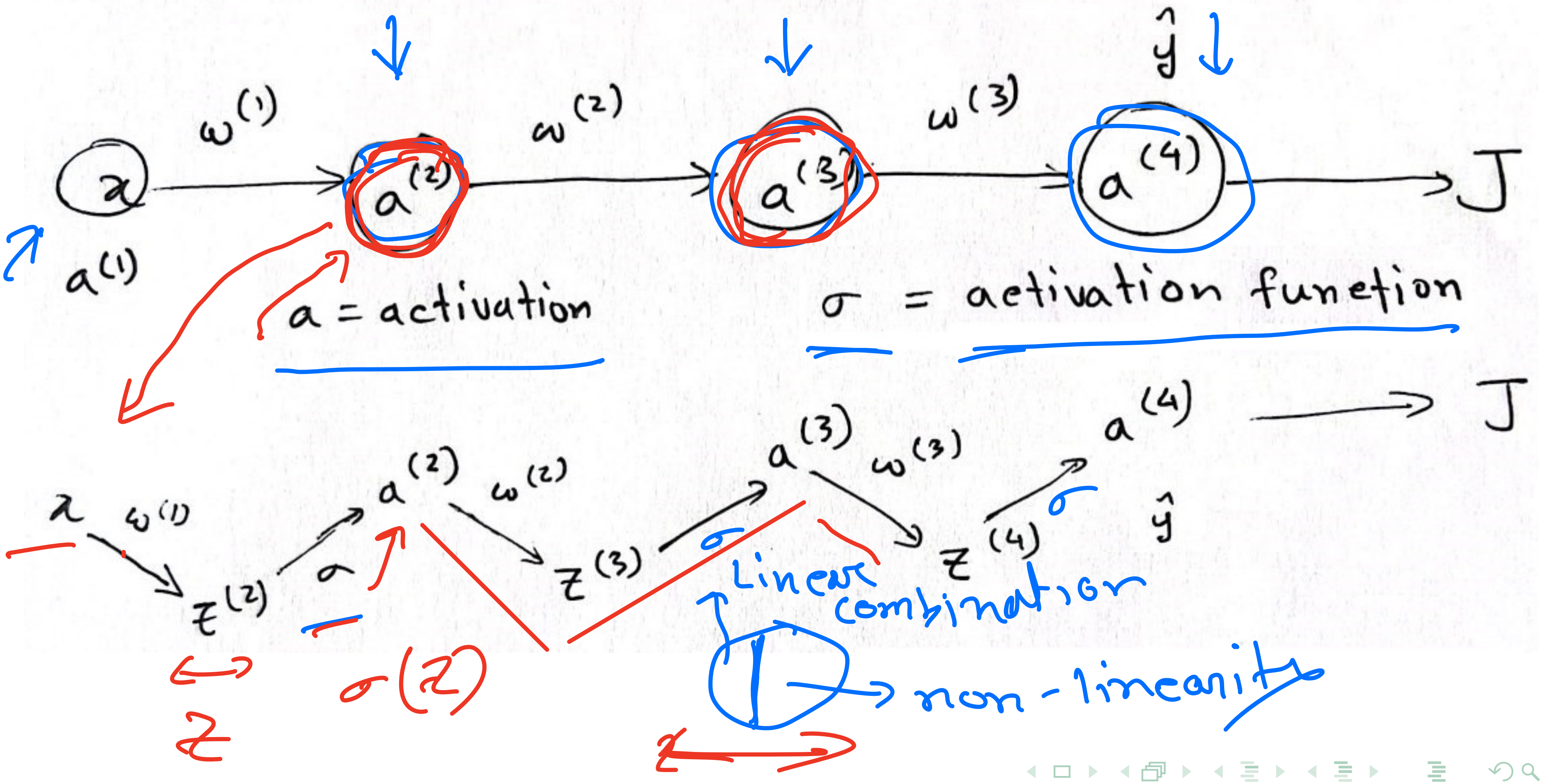
$$\|Ux\|_2 = \|x\|_2 , \quad U \text{ is orthogonal}$$

$U \in \mathbb{R}^{n \times n}$

$x \in \mathbb{R}^n$

Backpropagation (simplified)

Consider this network, with bias set to zero.



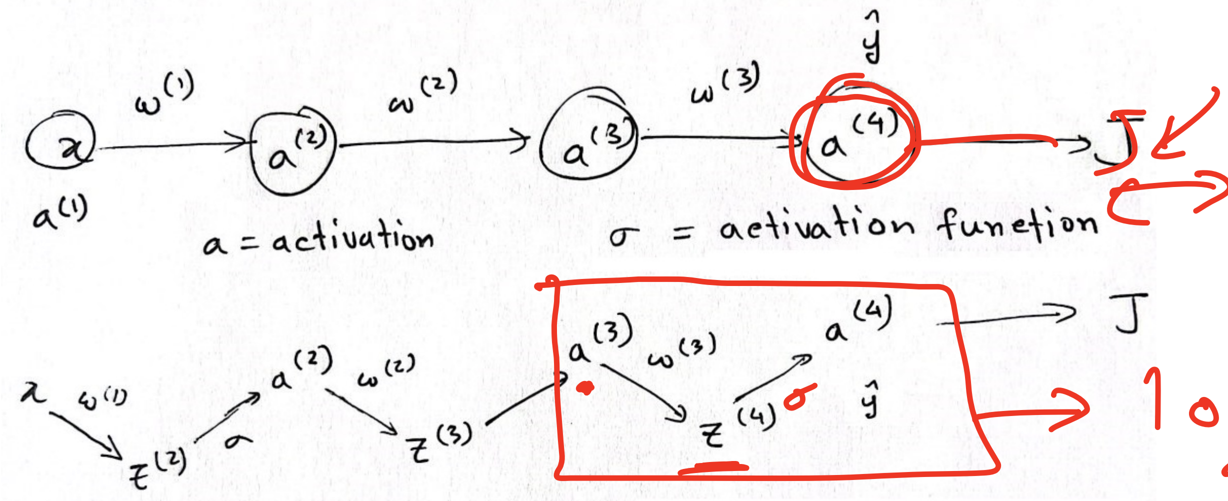
Backpropagation (simplified)

SGD

$w^{(1)} = 0.01$

$w^{(2)} = 0.1$

$w^{(3)} = -0.3$



logistic

$a^{(1)} = x$

$z^{(2)} = w^{(1)} a^{(1)}$

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

$z^{(3)} = w^{(2)} a^{(2)}$

$a^{(3)} = \sigma(z^{(3)})$

$z^{(4)} = w^{(3)} a^{(3)}$

$a^{(4)} = \sigma(z^{(4)})$

$\hat{y} = a^{(4)}$

Forward step

Consider a single data point

$$J = -\{y \ln \hat{y} + (1-y) \ln(1-\hat{y})\}$$

BCE

then,

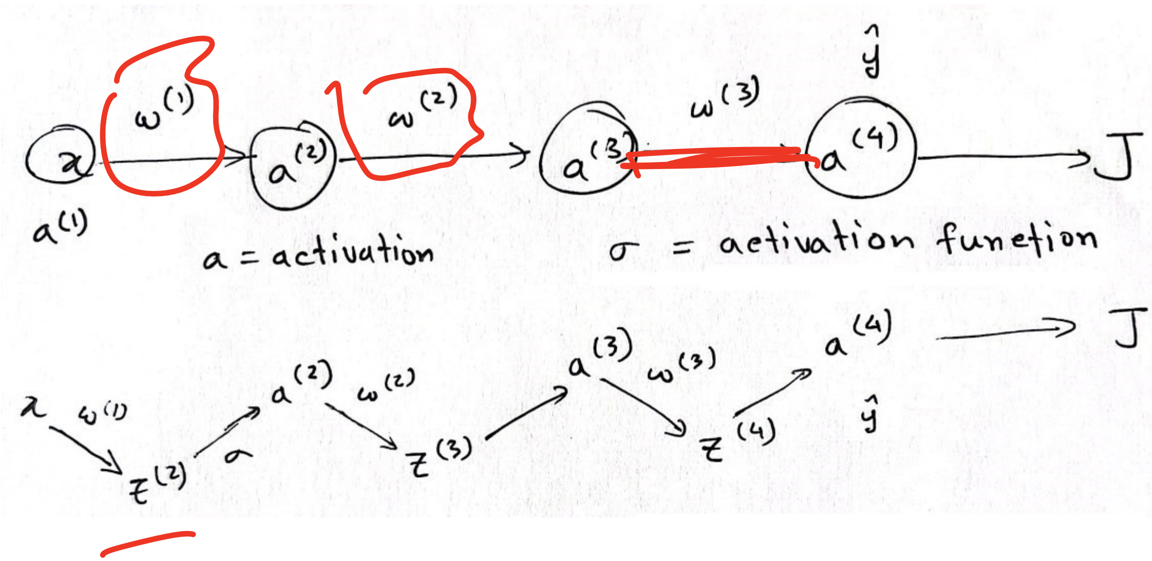
$$\frac{\partial J}{\partial w^{(3)}} = \frac{\partial J}{\partial a^{(4)}} \cdot \frac{\partial a^{(4)}}{\partial z^{(4)}} \cdot \frac{\partial z^{(4)}}{\partial w^{(3)}}$$

σ = sigmoid function

$$= -\{y - a^{(4)}\}\, a^{(3)}$$

[logistic regression equivanlent]

Backpropagation (simplified)



$\frac{\partial J}{\partial z^{(4)}} = \delta^{(4)}$

$a^{(1)} = x$

$z^{(2)} = w^{(1)} a^{(1)}$

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

$z^{(3)} = w^{(2)} a^{(2)}$

$a^{(3)} = \sigma(z^{(3)})$

$z^{(4)} = w^{(3)} a^{(3)}$

$a^{(4)} = \sigma(z^{(4)})$

$\hat{y} = a^{(4)}$

Forward step

Let's define, $\frac{\partial J}{\partial z^{(4)}} = \delta^{(4)}$

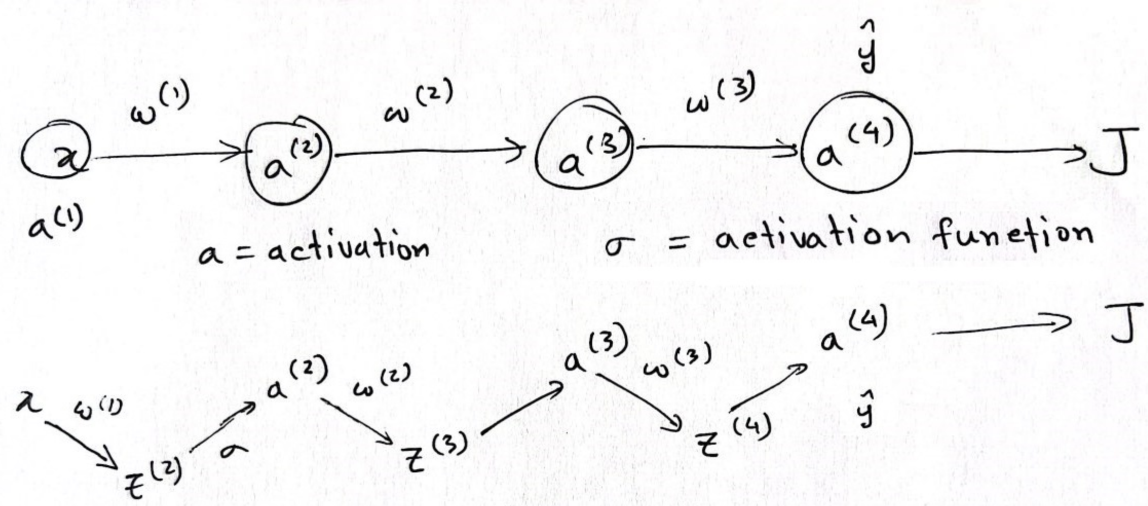
generally, $\frac{\partial J}{\partial z^{(l)}} = \delta^{(l)}$

$\frac{\partial J}{\partial z^{(l)}} = \delta^{(l)}$

only updates weight in (3) $w^{(3)}$

$$\boxed{\therefore \frac{\partial J}{\partial w^{(3)}} = -\{y - a^{(4)}\}\, a^{(3)}}$$

Backpropagation (simplified)



$$
\begin{aligned}
a^{(1)} &= x \\
z^{(2)} &= w^{(1)} a^{(1)} \\
a^{(2)} &= \sigma(z^{(2)}) \\
z^{(3)} &= w^{(2)} a^{(2)} \\
a^{(3)} &= \sigma(z^{(3)}) \\
z^{(4)} &= w^{(3)} a^{(3)} \\
a^{(4)} &= \sigma(z^{(4)}) \\
\hat{y} &= a^{(4)}
\end{aligned}
$$

Forward step

$$\frac{\partial J}{\partial w^{(2)}} = \boxed{\underbrace{\frac{\partial J}{\partial a^{(4)}} \cdot \frac{\partial a^{(4)}}{\partial z^{(4)}} \cdot \frac{\partial z^{(4)}}{\partial a^{(3)}} \cdot \frac{\partial a^{(3)}}{\partial z^{(3)}}}_{\frac{\partial J}{\partial z^{(3)}} = \delta^{(3)}}} \cdot \frac{\partial z^{(3)}}{\partial w^{(2)}}$$

$$\therefore \frac{\partial J}{\partial w^{(2)}} = \delta^{(3)} \; a^{(2)}$$

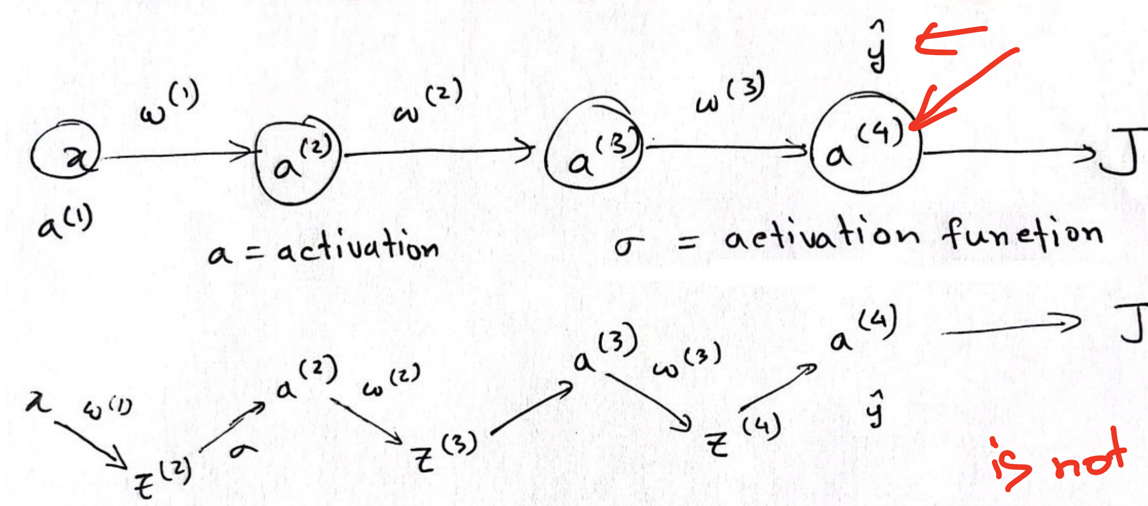
$$\frac{\partial}{\partial w^{(2)}} \left\{ w^{(2)} a^{(2)} \right\}$$

Similarly, $\dfrac{\partial J}{\partial w^{(1)}} = \delta^{(2)} \; a^{(1)}$

J varies with $w^{(1)}$

Generally, $\boxed{\dfrac{\partial J}{\partial w^{(l)}} = \delta^{(l+1)} \; a^{(l)}}$

Backpropagation (simplified)



Relu'
is not defined $x = 0.0001$

$$a^{(1)} = x$$
$$z^{(2)} = w^{(1)} a^{(1)}$$
$$a^{(2)} = \sigma(z^{(2)})$$
$$z^{(3)} = w^{(2)} a^{(2)}$$
$$a^{(3)} = \sigma(z^{(3)})$$
$$z^{(4)} = w^{(3)} a^{(3)}$$
$$a^{(4)} = \sigma(z^{(4)})$$
$$\hat{y} = a^{(4)}$$

Forward step

Network with L hidden layers, will have

error, $\delta^{(L)} = -\{y - \hat{a}^{(L)}\} = \dfrac{\partial J}{\partial z^{(L)}}$

known

For this network, $\delta^{(4)} = \dfrac{\partial J}{\partial z^{(4)}}$ is known

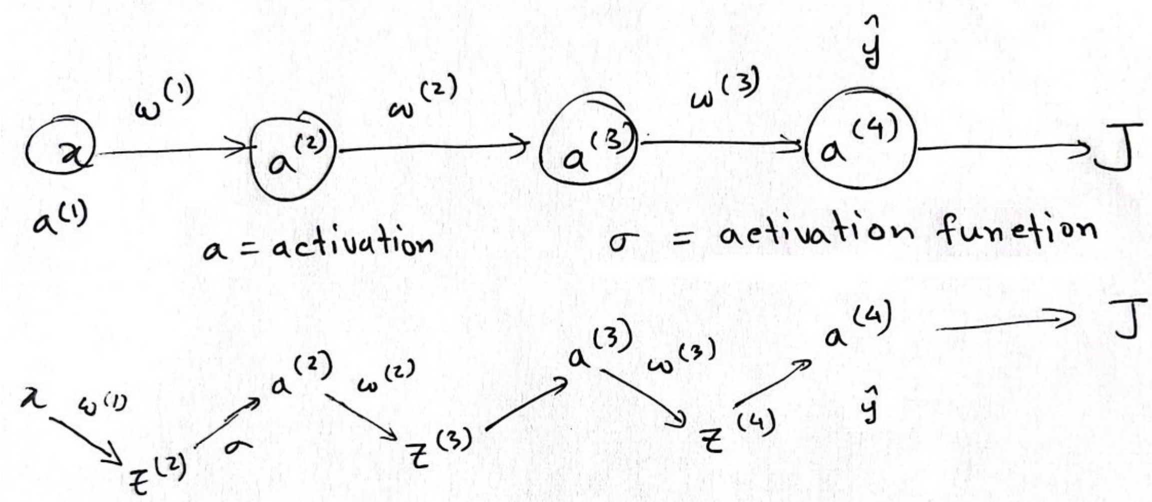
Now, $\delta^{(4)} = \dfrac{\partial J}{\partial a^{(4)}} \cdot \dfrac{\partial a^{(4)}}{\partial z^{(4)}}$

$$\delta^{(3)} = \frac{\partial J}{\partial z^{(3)}}$$

$$\delta^{(3)} = \underbrace{\frac{\partial J}{\partial a^{(4)}} \cdot \frac{\partial a^{(4)}}{\partial z^{(4)}}}_{\delta^{(4)}} \cdot \underbrace{\frac{\partial z^{(4)}}{\partial a^{(3)}}}_{w^{(3)}} \cdot \underbrace{\frac{\partial a^{(3)}}{\partial z^{(3)}}}_{\sigma'(z^{(3)})}$$

first time derivative of σ comes in

Backpropagation (simplified)



$$a^{(1)} = x$$
$$z^{(2)} = \omega^{(1)} a^{(1)}$$
$$a^{(2)} = \sigma(z^{(2)})$$
$$z^{(3)} = \omega^{(2)} a^{(2)}$$
$$a^{(3)} = \sigma(z^{(3)})$$
$$z^{(4)} = \omega^{(3)} a^{(3)}$$
$$a^{(4)} = \sigma(z^{(4)})$$
$$\hat{y} = a^{(4)}$$

Forward step

$$\therefore \delta^{(3)} = \delta^{(4)} \omega^{(3)} \sigma'(z^{(3)})$$

$$\boxed{\delta^{(l)} = \delta^{(l+1)} \omega^{(l)} \sigma'(z^{(l)})}$$

Combine with $\dfrac{\partial J}{\partial \omega^{(l)}} = \delta^{(l+1)} a^{(l)}$

A neural network

→ Conjugate Gradient

→ Quasi-Newton-method

u represents number of units or neurons per layer

'86

Rumelhart

backprop

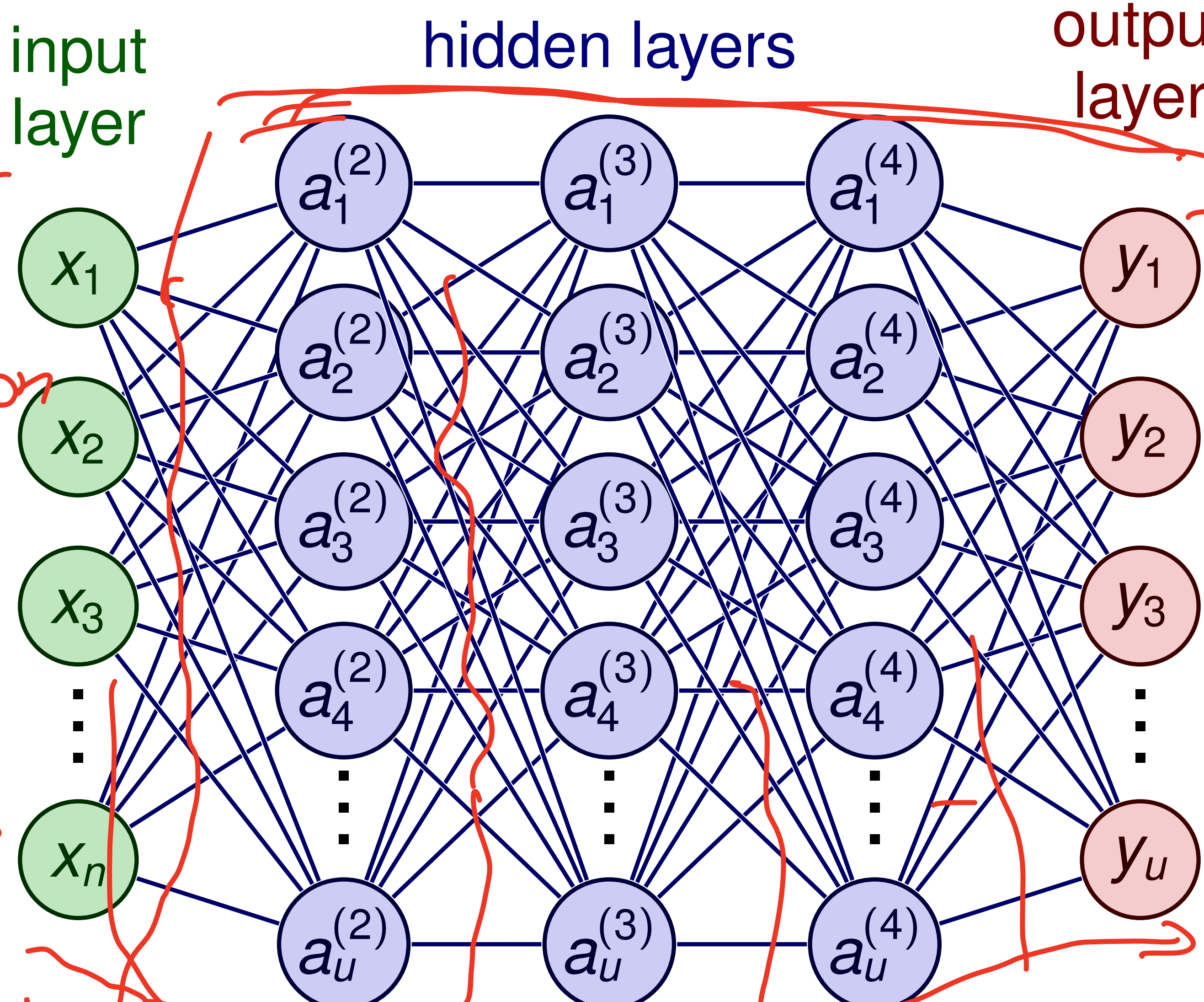
→ G. Hinton

weights

50-60s

KNN

Perceptron



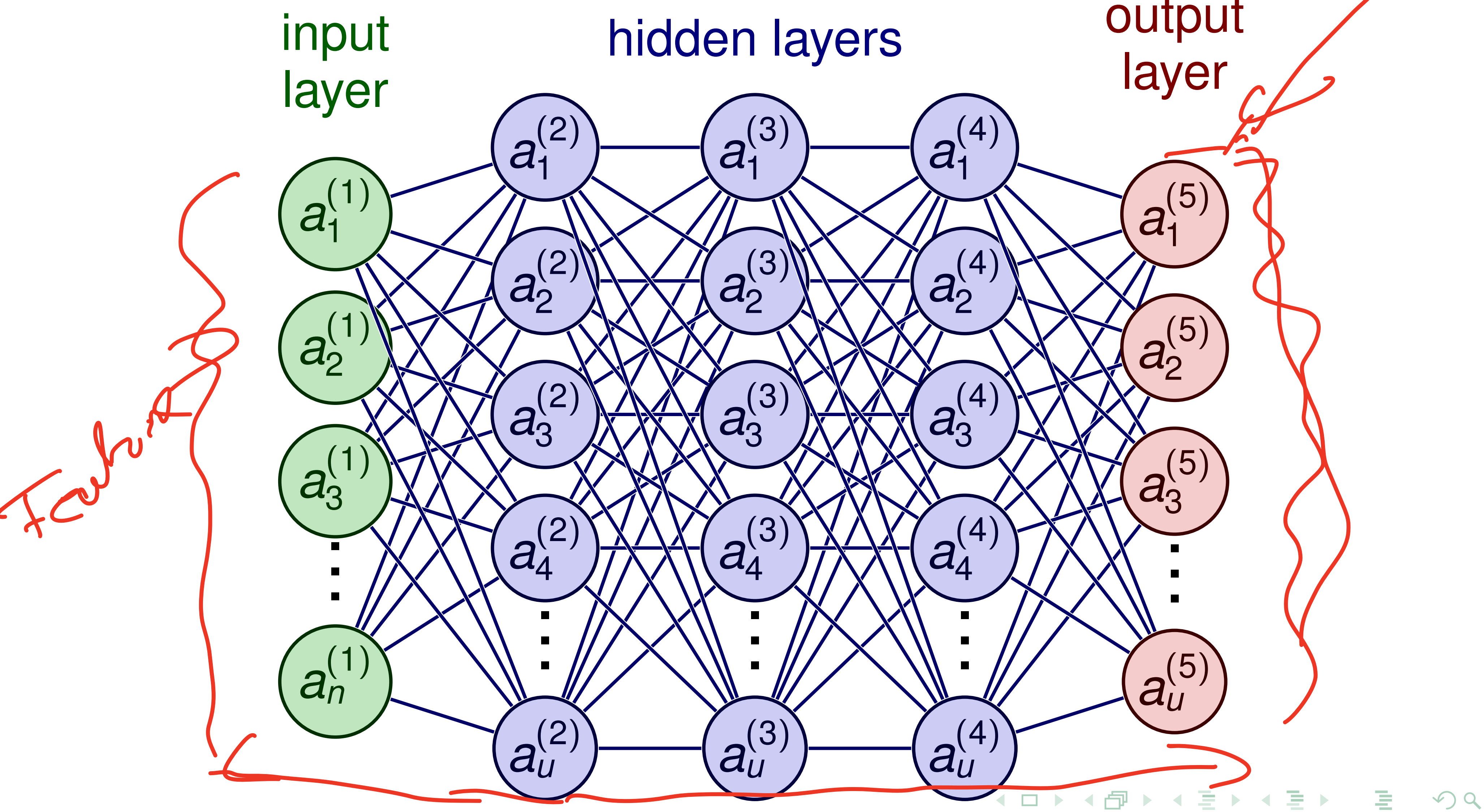
→ J

Back Prop

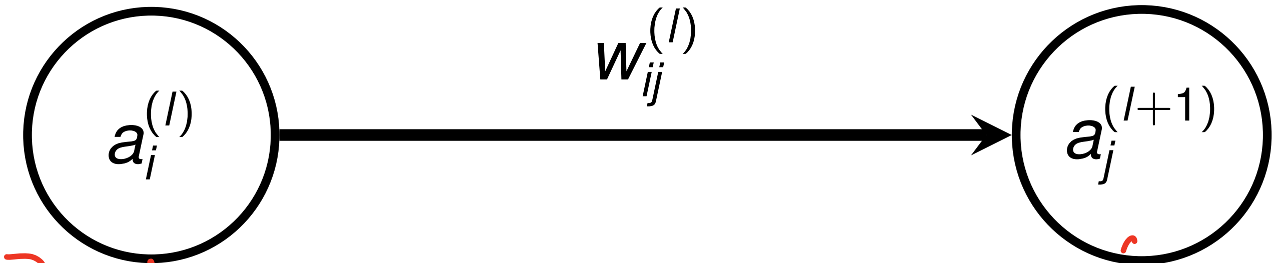
Forward

A neural network: Only activations

u represents number of units or neurons per layer



Forward Step: Calculate activations



$$a_j^{(l+1)} = \sigma(\sum_{i=0} w_{ij}^{(l)} a_i^{(l)})$$

Example : For the first neuron in the first hidden layer or second layer of the network, is

$$a_1^{(2)} = \sigma(\sum_{i=0} w_{i1}^{(1)} a_i^{(1)})$$

When i=0, $w01$ is taken into account. Similarly with every value of i you will progressively add the contribution of every neuron, to calculate the activation of the neuron of interest.

Backpropagation: Update model parameters to reduce cost

$$\delta^{(l)} = \delta^{(l+1)} w^{(l)} \sigma'(z^{(l)})$$

$$\frac{\partial J}{\partial w^{(l)}} = \delta^{(l+1)} a^{(l)}$$

$\delta^{(L)}$ is known for the L-th layer.

Backpropagation: Update model parameters to reduce cost

w - Matrix

v - vector

$$\frac{\partial J}{\partial w_{ij}^{(l)}} = \delta_j^{(l+1)} a_i^{(l)}$$

$$\vec{\delta^{(l)}} = W^{(l)} \vec{\delta^{(l)}} \odot \sigma'(\vec{z^{(l)}})$$

matrix vector vector

$\odot$ is elementwise product or Hadamard product

vector $\odot$ vector

= vector