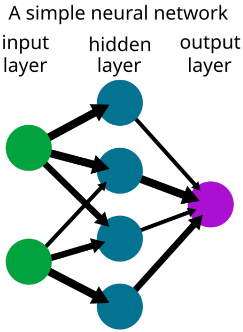
Neural Network: Theory and Practice

Jephian C.-H. Lin 林晉宏

Department of Applied Mathematics, National Sun Yat-sen University

February 10, 2025
The 2nd Sizihwan Combinatorics Conference

What is an artificial neural network?

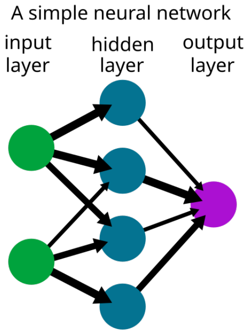


(Source: Wikipedia of Neural network)

keep improving itself

Not easy to find learning resources for mathematicians.

What is an artificial neural network?

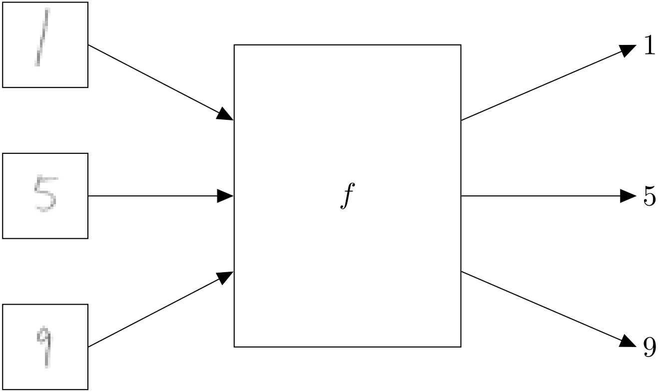


(Source: Wikipedia of Neural network)

keep improving itself

Not easy to find learning resources for mathematicians.

Classification task



Input: Flattened vector

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	26	35	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	72	71	47	59	46	47	63	41	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	2	97	47	45	27	12	6	2	1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	35	78	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	60	50	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	108	29	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	29	57	73	74	69	28	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	1	8	25	58	60	6	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	48	47	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	22	82	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	46	84	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	9	6	0	0	0	9	99	24	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	14	59	40	24	44	79	34	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	6	33	54	34	8	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

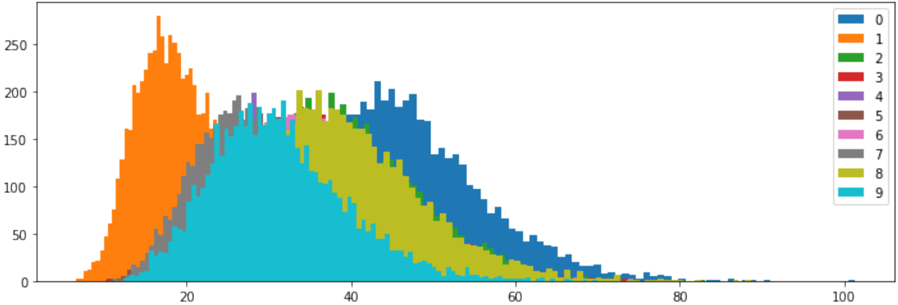
$\text{pic}_i \in \mathbb{R}^{28\times 28}$

↓

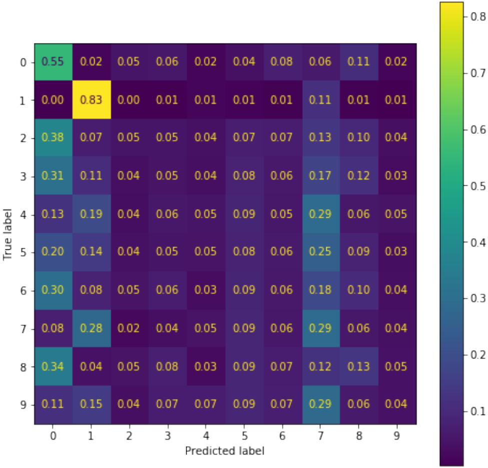
0 ... 0 0 0 0 0 72 71 47 59 46 47 63 41 1 0 0 ... 0

$\mathbf{x}_i^\top \in \mathbb{R}^{784}$

- $f(\mathbf{x}_i^\top) = 1$ (accuracy 10%)
- $f(\mathbf{x}_i^\top) = \mathbf{x}_i^\top \mathbf{1} \to$ guess the color by ink amount (accuracy 22%)



Confusion matrix of the ink-guessing strategy

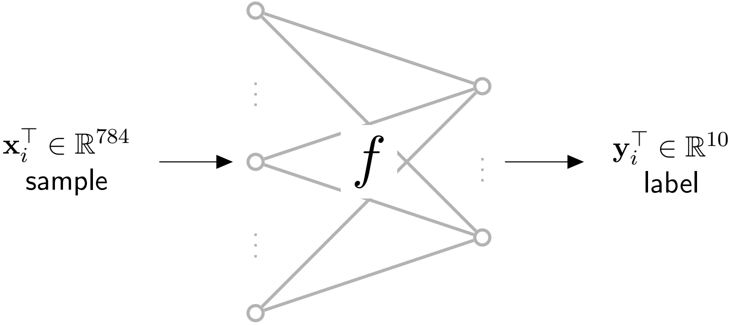


Output: One-hot encoding

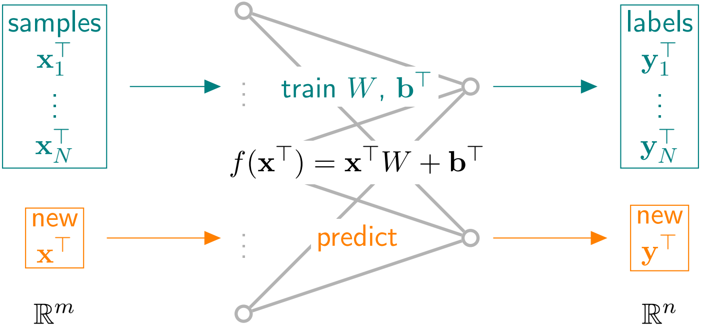
$$
\begin{array}{ccl}
1 & \rightarrow & \begin{bmatrix}0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\end{bmatrix} \\
\vdots & & \qquad\qquad\qquad\vdots \\
5 & \rightarrow & \begin{bmatrix}0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0\end{bmatrix} \quad \in \quad \mathbb{R}^{10} \\
\vdots & & \qquad\qquad\qquad\vdots \\
9 & \rightarrow & \begin{bmatrix}0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1\end{bmatrix}
\end{array}
$$

One-hot encoding ensures the labels of $0, \ldots, 9$ are mutually equal-distanced.

A layer ~ a transformation



Linear model



Problem

Find $W \in \mathbb{R}^{m\times n}$ *and* $\mathbf{b}^\top \in \mathbb{R}^n$ *that minimizes the cost function*

$$\frac{1}{N}\sum_{i=1}^{N} \|f(\mathbf{x}_i^\top) - \mathbf{y}_i\|^2.$$

Solve the optimization problem — by projection

Let

$$X = \begin{bmatrix} - & \mathbf{x}_1 & - \\ & \vdots & \\ - & \mathbf{x}_N & - \end{bmatrix} \in \mathbb{R}^{N\times m} \text{ and } Y = \begin{bmatrix} - & \mathbf{y}_1 & - \\ & \vdots & \\ - & \mathbf{y}_N & - \end{bmatrix} \in \mathbb{R}^{N\times n}.$$

The cost function becomes

$$\frac{1}{N}\|XW + \mathbf{1}_N\mathbf{b}^\top - Y\|^2 = \frac{1}{N}\|\hat{X}\hat{W} - Y\|^2,$$

where

$$\hat{X} = \begin{bmatrix} \mathbf{1}_N & X \end{bmatrix} \text{ and } \hat{W} = \begin{bmatrix} \mathbf{b}^\top \\ W \end{bmatrix}.$$

The minimum uniquely occurs at $\hat{W} = (\hat{X}^\top\hat{X})^{-1}\hat{X}^\top Y$.

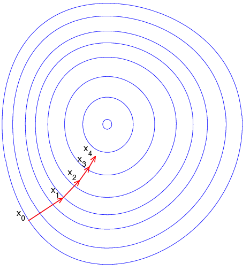
Memory consuming. Hard to update for new data.

Solve the optimization problem — by gradient descent

Input a scalar-valued function f and a learning rate $\alpha = 0.1$

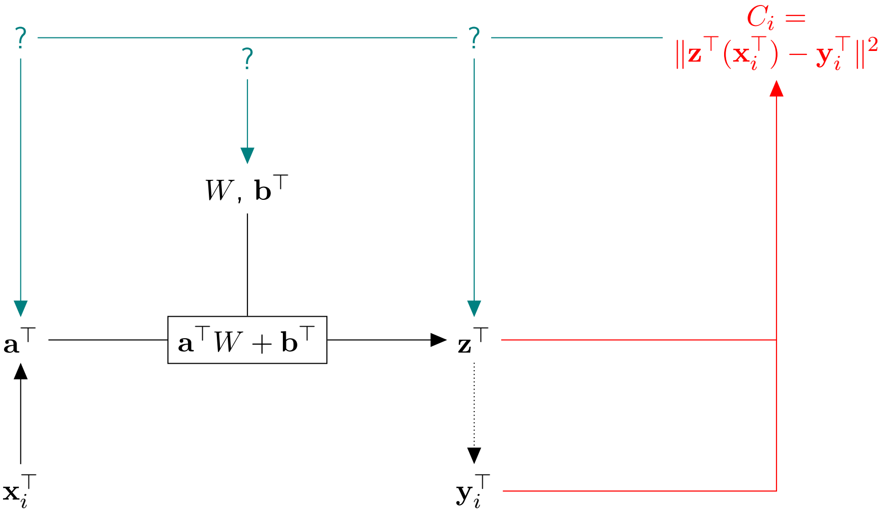
Output a point $\mathbf{x}$ such that $f(\mathbf{x})$ is locally minimized, hopefully

Steps
1. Initiate $\mathbf{x}$ randomly.
2. Compute $\mathbf{v} = \nabla f(\mathbf{x})$.
3. $\mathbf{x} \leftarrow \mathbf{x} - \alpha\mathbf{v}$.
4. Repeat Step 2 $\sim$ Step 3 several iterations.



(Source: Wikipedia of Gradient descent)

Linear model



Derivative and gradient

Derivative is a linear operator
sending a direction to the directional derivative.

$$\frac{\partial f}{\partial \mathbf{x}}(\Delta_{\mathbf{x}}) = \lim_{t\to 0} \frac{f(\mathbf{x}+t\Delta_{\mathbf{x}}) - f(\mathbf{x})}{t} \overset{\text{if } f \text{ scalar-valued}}{=\!=\!=\!=} \underbrace{\nabla_{\mathbf{x}} f}_{\text{gradient}} \cdot \Delta_{\mathbf{x}}.$$

Example

Let $C_i = \|\mathbf{z}^\top(\mathbf{x}_i^\top) - \mathbf{y}_i^\top\|$. Then

$$\begin{aligned}
\frac{\partial C_i}{\partial \mathbf{z}^\top}(\Delta_{\mathbf{z}^\top}) &= \lim_{t\to 0} \frac{\|\mathbf{z}^\top + t\Delta_{\mathbf{z}^\top} - \mathbf{y}_i^\top\|^2 - \|\mathbf{z}^\top - \mathbf{y}_i^\top\|^2}{t} \\
&= \lim_{t\to 0} \frac{(\mathbf{z}^\top + t\Delta_{\mathbf{z}^\top} - \mathbf{y}_i^\top)\cdot(\mathbf{z}^\top + t\Delta_{\mathbf{z}^\top} - \mathbf{y}_i^\top) - \|\mathbf{z}^\top - \mathbf{y}_i^\top\|^2}{t} \\
&= 2(\mathbf{z}^\top - \mathbf{y}_i^\top)\cdot \Delta_{\mathbf{z}^\top}.
\end{aligned}$$

Derivative is a linear operator
sending a direction to the directional derivative.

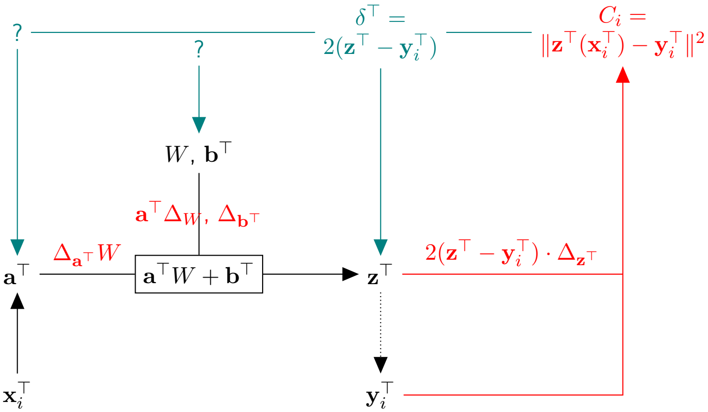
$$\frac{\partial f}{\partial \mathbf{x}}(\Delta_{\mathbf{x}}) = \lim_{t\to 0} \frac{f(\mathbf{x}+t\Delta_{\mathbf{x}}) - f(\mathbf{x})}{t} \overset{\text{if } f \text{ scalar-valued}}{=\!=\!=\!=} \underbrace{\nabla_{\mathbf{x}} f}_{\text{gradient}} \cdot \Delta_{\mathbf{x}}.$$

Example

Let $f(\mathbf{a}^\top) = \mathbf{a}^\top W + \mathbf{b}^\top$. Then

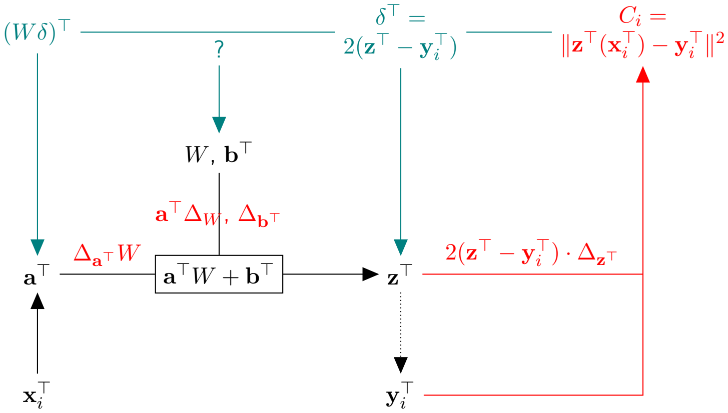
$$\frac{\partial f}{\partial \mathbf{a}^\top}(\Delta_{\mathbf{a}^\top}) = \Delta_{\mathbf{a}^\top} W, \quad \frac{\partial f}{\partial W}(\Delta_W) = \mathbf{a}^\top \Delta_W, \quad \frac{\partial f}{\partial \mathbf{b}^\top}(\Delta_{\mathbf{b}^\top}) = \Delta_{\mathbf{b}^\top}.$$

Linear model



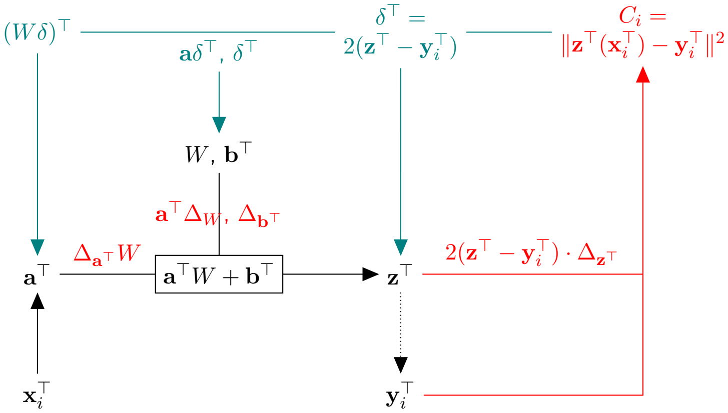
Linear model

$$\frac{\partial C_i}{\partial \mathbf{a}} = \Delta_{\mathbf{a}^\top} W \delta = (W\delta)^\top \cdot \Delta_{\mathbf{a}^\top}$$

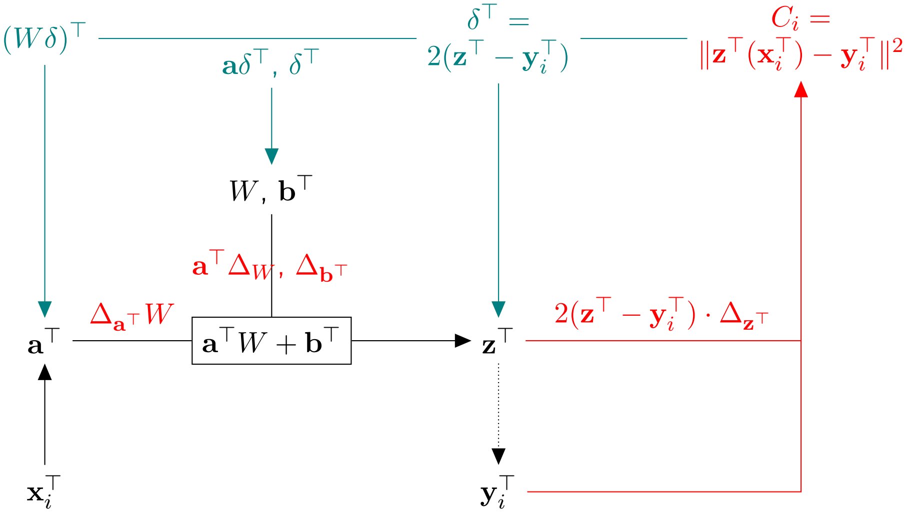


Linear model

$$\frac{\partial C_i}{\partial W} = \mathbf{a}^\top \Delta_W \delta = (\mathbf{a}\delta^\top) \cdot \Delta_W, \quad \frac{\partial C_i}{\partial \mathbf{b}^\top} = \delta^\top \cdot \Delta_{\mathbf{b}^\top}$$



Linear model

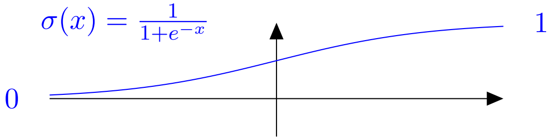


Activation function

Two linear layers is the same as one layer.

$$(\mathbf{a}^\top W_1 + \mathbf{b}_1^\top)W_2 + \mathbf{b}_2^\top = \mathbf{a}^\top W_1 W_2 + (\mathbf{b}_1^\top W_2 + \mathbf{b}_2^\top)$$

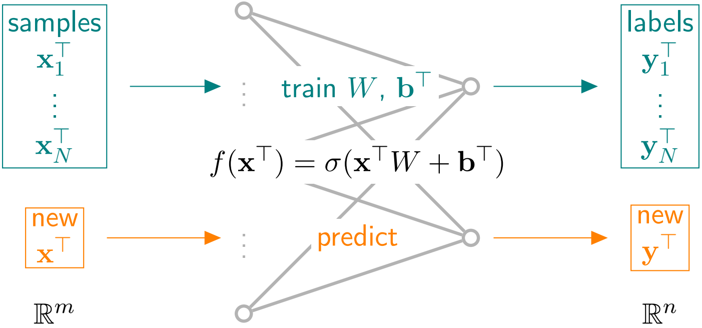
An activation function adds nonlinearity to the function.



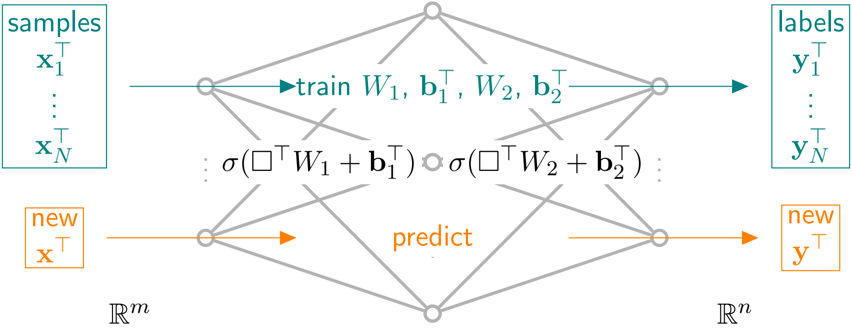
sigmoid function

Define $\sigma(x_1, \ldots, x_m) = (\sigma(x_1), \ldots, \sigma(x_m))$.
So $\frac{\partial \sigma}{\partial \mathbf{x}}(\Delta_{\mathbf{x}}) = \Delta_{\mathbf{x}} \circ (\sigma'(x_1), \ldots, \sigma'(x_m)) = \Delta_{\mathbf{x}} \circ \sigma'(\mathbf{x})$.

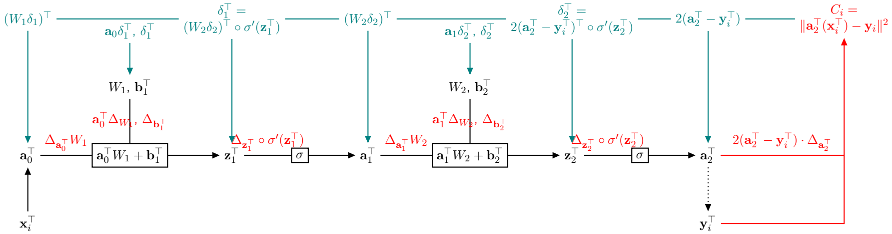
Neural network: Single-layer perceptron



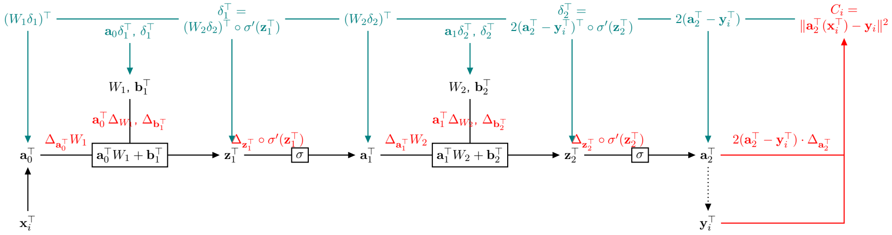
Neural network: Multilayer perceptron



Neural network: Multilayer perceptron

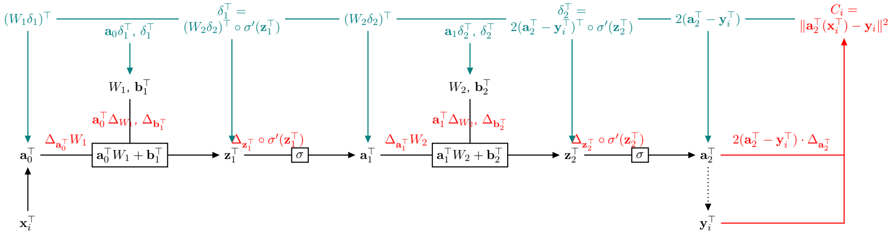


Some examples



$(W_1\delta_1)^\top$
$\delta_1^\top =$
$(W_2\delta_2)^\top \circ \sigma'(\mathbf{z}_1^\top)$
$(W_2\delta_2)^\top$
$\delta_2^\top =$
$2(\mathbf{a}_2^\top - \mathbf{y}_i^\top)^\top \circ \sigma'(\mathbf{z}_2^\top)$
$2(\mathbf{a}_2^\top - \mathbf{y}_i^\top)$
$C_i =$
$\|\mathbf{a}_2^\top(\mathbf{x}_i^\top) - \mathbf{y}_i\|^2$
$\mathbf{a}_0\delta_1^\top,\ \delta_1^\top$
$\mathbf{a}_1\delta_2^\top,\ \delta_2^\top$
$W_1,\ \mathbf{b}_1^\top$
$W_2,\ \mathbf{b}_2^\top$
$\mathbf{a}_0^\top\Delta_{W_1},\ \Delta_{\mathbf{b}_1^\top}$
$\mathbf{a}_1^\top\Delta_{W_2},\ \Delta_{\mathbf{b}_2^\top}$
$\Delta_{\mathbf{a}_0^\top} W_1$
$\Delta_{\mathbf{z}_1^\top} \circ \sigma'(\mathbf{z}_1^\top)$
$\Delta_{\mathbf{a}_1^\top} W_2$
$\Delta_{\mathbf{z}_2^\top} \circ \sigma'(\mathbf{z}_2^\top)$
$2(\mathbf{a}_2^\top - \mathbf{y}_i^\top) \cdot \Delta_{\mathbf{a}_2^\top}$
$\mathbf{a}_0^\top$
$\mathbf{a}_0^\top W_1 + \mathbf{b}_1^\top$
$\mathbf{z}_1^\top$
σ
$\mathbf{a}_1^\top$
$\mathbf{a}_1^\top W_2 + \mathbf{b}_2^\top$
$\mathbf{z}_2^\top$
σ
$\mathbf{a}_2^\top$
$\mathbf{x}_i^\top$
$\mathbf{y}_i^\top$

Thanks!


$(W_1\delta_1)^\top$
$\delta_1^\top = (W_2\delta_2)^\top \circ \sigma'(\mathbf{z}_1^\top)$
$(W_2\delta_2)^\top$
$\delta_2^\top = 2(\mathbf{a}_2^\top - \mathbf{y}_i^\top)^\top \circ \sigma'(\mathbf{z}_2^\top)$
$2(\mathbf{a}_2^\top - \mathbf{y}_i^\top)$
$C_i = \|\mathbf{a}_2^\top(\mathbf{x}_i^\top) - \mathbf{y}_i\|^2$
$\mathbf{a}_0\delta_1^\top,\ \delta_1^\top$
$\mathbf{a}_1\delta_2^\top,\ \delta_2^\top$
$W_1,\ \mathbf{b}_1^\top$
$W_2,\ \mathbf{b}_2^\top$
$\mathbf{a}_0^\top\Delta_{W_1},\ \Delta_{\mathbf{b}_1^\top}$
$\mathbf{a}_1^\top\Delta_{W_2},\ \Delta_{\mathbf{b}_2^\top}$
$\Delta_{\mathbf{a}_0^\top}W_1$
$\Delta_{\mathbf{z}_1^\top} \circ \sigma'(\mathbf{z}_1^\top)$
$\Delta_{\mathbf{a}_1^\top}W_2$
$\Delta_{\mathbf{z}_2^\top} \circ \sigma'(\mathbf{z}_2^\top)$
$2(\mathbf{a}_2^\top - \mathbf{y}_i^\top) \cdot \Delta_{\mathbf{a}_2^\top}$
$\mathbf{a}_0^\top$
$\mathbf{a}_0^\top W_1 + \mathbf{b}_1^\top$
$\mathbf{z}_1^\top$
σ
$\mathbf{a}_1^\top$
$\mathbf{a}_1^\top W_2 + \mathbf{b}_2^\top$
$\mathbf{z}_2^\top$
σ
$\mathbf{a}_2^\top$
$\mathbf{x}_i^\top$
$\mathbf{y}_i^\top$

Thanks!

Michael A. Nielsen.
Neural Networks and Deep Learning.
Determination Press, 2015.
http://neuralnetworksanddeeplearning.com/.