Notes of Caltech ML Lecture:

Neural Network

Trần Trung Kiên

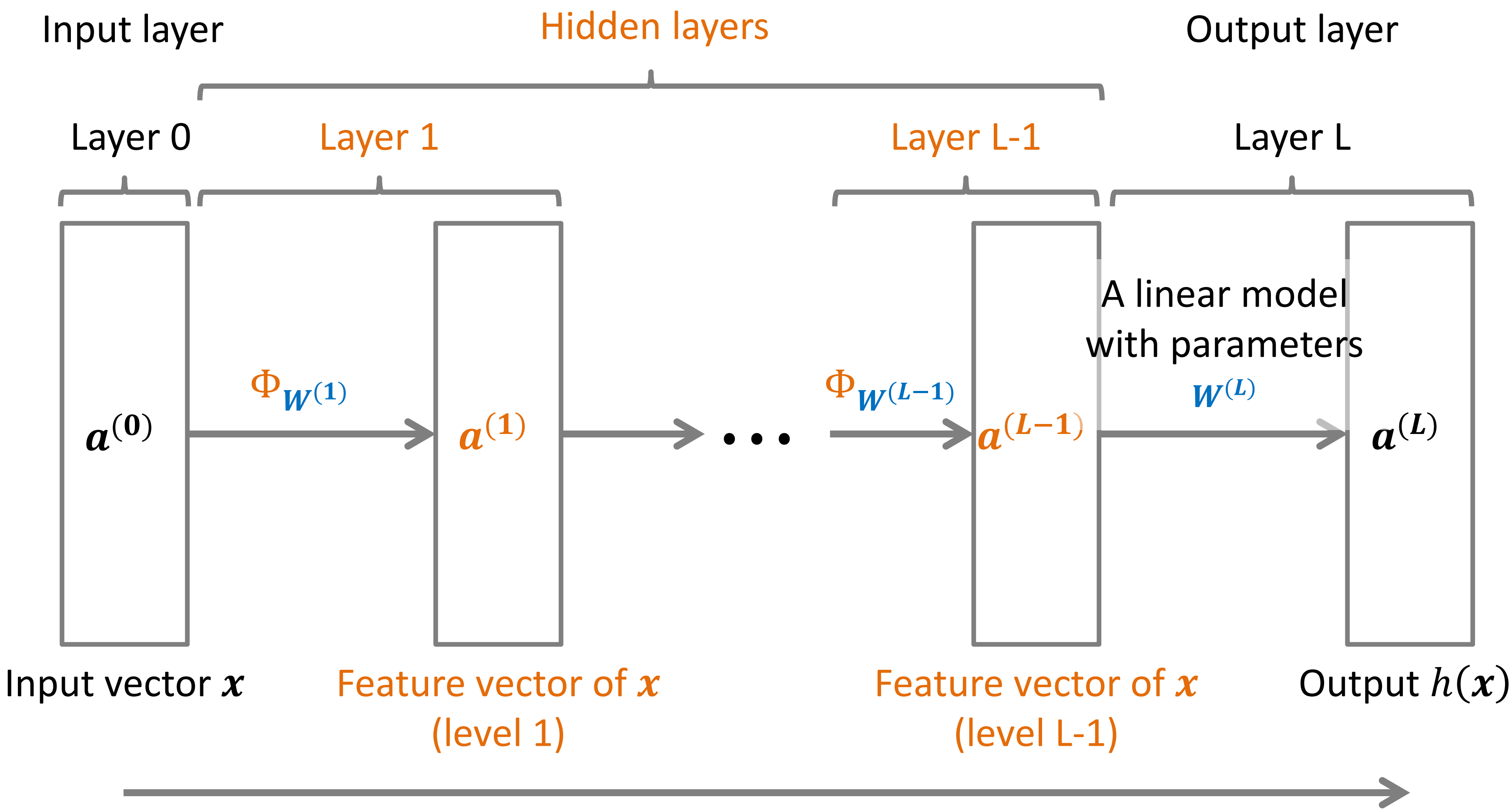
ttkien@fit.hcmus.edu.vn

KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

Overview

- Hypothesis set
- Learning algorithm

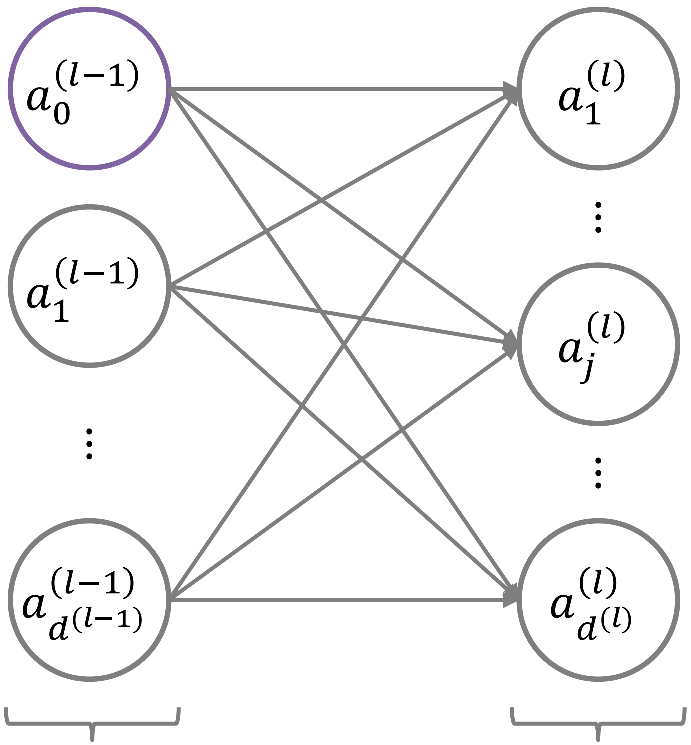
Hypothesis set



The process of computing $h(\boldsymbol{x})$ from $\boldsymbol{x}$ is call forward-propagation

Neural Network vs Linear Regression, Logistic Regression?

How is $a^{(l)}$ computed from $a^{(l-1)}$?

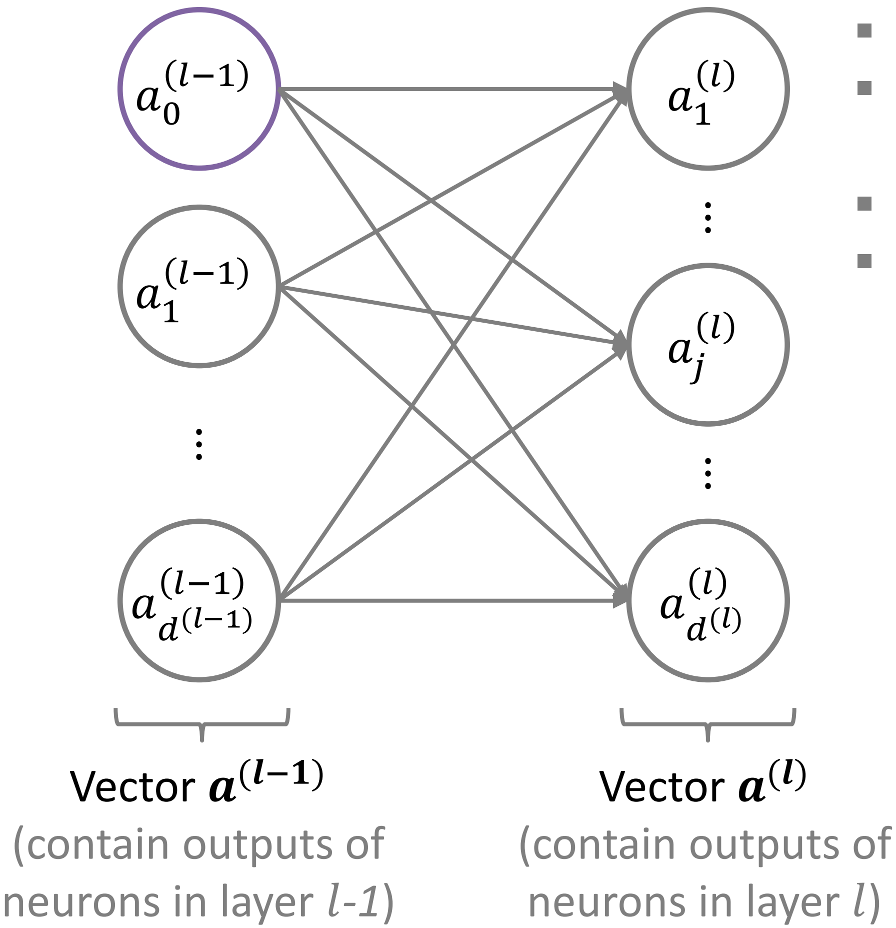


Vector $\boldsymbol{a}^{(l-1)}$
(contain outputs of neurons in layer l-1)

Vector $\boldsymbol{a}^{(l)}$
(contain outputs of neurons in layer l)

- Each node is a neuron
- Value inside a node is the output of this neuron
- Each edge is a weight (parameter)
- **The output of a neuron is computed as follows**: (1) compute the weighted sum of neurons' outputs in the previous layer, (2) pass the result through a nonlinear function – called activation function (e.g. logistic)
  - $w_{ij}^{(l)}$: the weight corresponding to the edge from neuron i in layer $l-1$ to neuron j in layer l $(1 \leq l \leq L,\ 0 \leq i \leq d^{(l-1)}, 1 \leq j \leq d^{(l)})$ [Check: blackboard]
  - $a_j^{(l)} = \theta\left(s_j^{(l)}\right)$ with $s_j^{(l)} =$ and θ is some activation function

How is $a^{(l)}$ computed from $a^{(l-1)}$?



- Each node is a neuron
- Value inside a node is the output of this neuron
- Each edge is a weight (parameter)
- **The output of a neuron is computed as follows**: (1) compute the weighted sum of neurons' outputs in the previous layer, (2) pass the result through a nonlinear function – called activation function (e.g. logistic)
  - $w_{ij}^{(l)}$: the weight corresponding to the edge from neuron i in layer $l-1$ to neuron j in layer l $(1 \le l \le L,\ 0 \le i \le d^{(l-1)}, 1 \le j \le d^{(l)})$ [Check: blackboard]
  - $a_j^{(l)} = \theta\left(s_j^{(l)}\right)$ with $s_j^{(l)} = \sum_{i=0}^{d^{(l-1)}} w_{ij}^{(l)} a_i^{(l-1)}$ and θ is some activation function

Activation function

- Common activation functions:

Name	$\theta(s)$	Range	$\theta'(s)$
Logistic	$\theta(s) = \frac{1}{1+e^{-s}}$	$[0, 1]$	
Tanh	$\theta(s) = \frac{e^{s}-e^{-s}}{e^{s}+e^{-s}}$	$[-1, 1]$	
Rectified linear	$\theta(s) = \max(0, s)$	$[0, \infty)$	
Linear	$\theta(s) = s$		

- θ for output layer: dictated by the problem
  - Regression: θ often is linear
  - Binary classification: θ often is logistic
- θ for hidden layers: hyper-parameter

Hypothesis set

After choosing a net architecture (# hidden layers, # neurons / hidden layer, θ), we will have a hypothesis set

1 specific $\boldsymbol{w} = \left\{ w_{ij}^{(l)} \right\} \leftrightarrow$ 1 hypothesis h

Overview

- ☐ Hypothesis set
- ☐ Learning algorithm

Learning algorithm

- Given:
  - Training data $\{(\boldsymbol{x}^{(1)}, y^{(1)}), \ldots, (\boldsymbol{x}^{(N)}, y^{(N)})\}$
    $\boldsymbol{x}^{(n)} \in \{1\} \times \mathbb{R}^d, y^{(n)} \in ?$
  - Hypothesis set $\mathcal{H} = \{h(\boldsymbol{x})\}$ corresponding to a net architecture

- $\boldsymbol{w} = ?$
  - Define $E_{in}(\boldsymbol{w})$

$$E_{in}(\boldsymbol{w}) = \frac{1}{\mathrm{N}} \sum_{n=1}^{N} \underbrace{e\left(h\left(x^{(n)}\right), y^{(n)}\right)}_{?}$$

  - $\min E_{in}(\boldsymbol{w})$

$\min E_{in}(\boldsymbol{w})$

(Batch) Gradient Descent (BGD)

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \alpha \nabla E_{in}(w)$$

$$\nabla E_{in}(w) = \frac{1}{N} \sum_{n=1}^{N} \nabla e\left(h\left(\boldsymbol{x}^{(n)}\right), y^{(n)}\right)$$

To take a step, BGD needs to go through N training examples

→ Slow when N is large

$\min E_{in}(\boldsymbol{w})$

Stochastic Gradient Descent (SGD)

Idea: use a minibatch of B ($B \ll N$) examples to estimate ∇E_{in}

Algorithm:

1. Initialize $\boldsymbol{w}$
2. While termination criterion is not satisfied
   a. Shuffle the order of training examples
   b. For minibatch $b = 1, \dots, \frac{N}{B}$:

1 epoch

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \alpha \frac{1}{B} \sum_{n=(b-1)B+1}^{bB} \nabla e\left(h\left(\boldsymbol{x}^{(n)}\right), y^{(n)}\right)$$

Stochasticity also helps optimization a little bit: blackboard

Compute ∇e

To apply SGD or BGD, we need to compute ∇e (it contains $\frac{\partial e}{\partial w_{ij}^{(l)}} \forall l, i, j$)

Back-propagation **algorithm:**

help us compute $\frac{\partial e}{\partial w_{ij}^{(l)}} \forall l, i, j$ efficiently

Review of chain rule

1 path

$$x \rightarrow y = f_1(x) \rightarrow z = f_2(y)$$

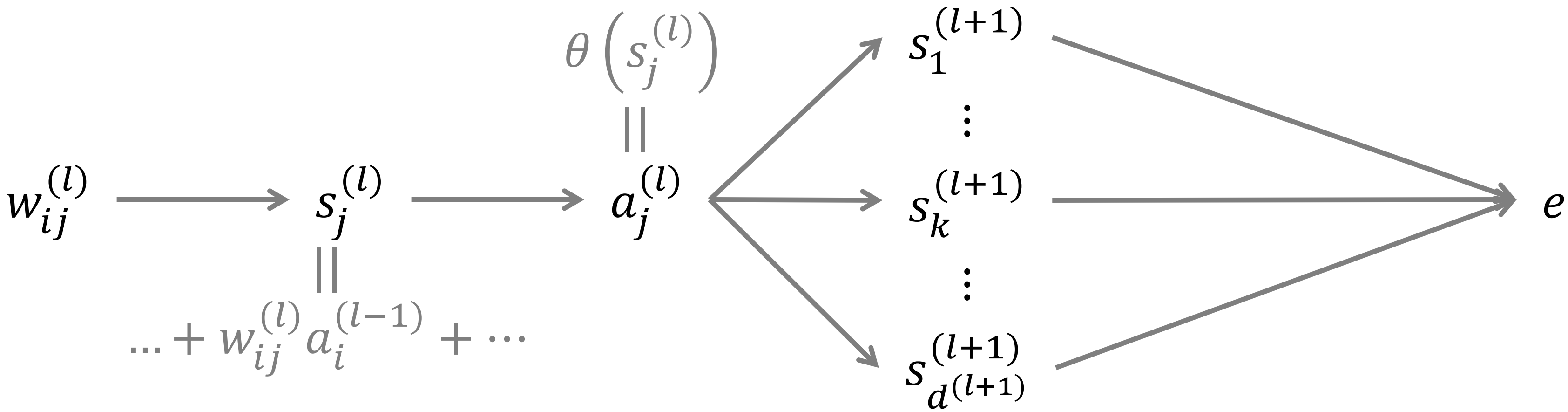
$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y} \frac{\partial y}{\partial x}$$

More than 1 path

$$x \rightarrow y_1 = f_1(x) \rightarrow z = f_3(y_1, y_2)$$

$$\searrow y_2 = f_2(x) \nearrow$$

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y_1} \frac{\partial y_1}{\partial x} + \frac{\partial z}{\partial y_2} \frac{\partial y_2}{\partial x}$$



$$\frac{\partial e}{\partial w_{ij}^{(l)}} = \underbrace{\frac{\partial e}{\partial s_j^{(l)}}}_{\delta_j^{(l)}} \underbrace{\frac{\partial s_j^{(l)}}{\partial w_{ij}^{(l)}}}_{a_i^{(l-1)}}$$

$$\underbrace{\frac{\partial e}{\partial a_j^{(l)}}} \underbrace{\frac{\partial a_j^{(l)}}{\partial s_j^{(l)}}}_{\theta'\left(s_j^{(l)}\right)}$$

$$\sum_{k=1}^{d^{(l+1)}} \underbrace{\frac{\partial e}{\partial s_k^{(l+1)}}}_{\delta_k^{(l+1)}} \underbrace{\frac{\partial s_k^{(l+1)}}{\partial a_j^{(l)}}}_{w_{jk}^{(l+1)}}$$

$$
\begin{tikzcd}
 & & \theta\left(s_j^{(l)}\right) \arrow[d, phantom, "\parallel"] & s_1^{(l+1)} \arrow[ddr] & \\
 & & & \vdots & \\
w_{ij}^{(l)} \arrow[r] & s_j^{(l)} \arrow[r] \arrow[d, phantom, "\parallel"] & a_j^{(l)} \arrow[uur] \arrow[r] \arrow[ddr] & s_k^{(l+1)} \arrow[r] & e \\
 & \dots + w_{ij}^{(l)} a_i^{(l-1)} + \cdots & & \vdots & \\
 & & & s_{d^{(l+1)}}^{(l+1)} \arrow[uur] &
\end{tikzcd}
$$

$$
\frac{\partial e}{\partial w_{ij}^{(l)}} = \underbrace{\frac{\partial e}{\partial s_j^{(l)}}}_{\delta_j^{(l)}} \underbrace{\frac{\partial s_j^{(l)}}{\partial w_{ij}^{(l)}}}_{a_i^{(l-1)}}
$$

$$
\delta_j^{(l)} = \theta'\left(s_j^{(l)}\right) \sum_{k=1}^{d^{(l+1)}} w_{jk}^{(l+1)} \delta_k^{(l+1)}
$$

To compute $\frac{\partial e}{\partial w_{ij}^{(l)}}$ $\forall l, i, j$, we need to compute:

1. $a_i^{(l-1)}$ $\forall l, i$: $\boldsymbol{x} = \boldsymbol{a}^{(0)} \to \boldsymbol{a}^{(1)} \to \cdots \to \boldsymbol{a}^{(L)}$
2. $\delta_j^{(l)}$ $\forall l, j$: $\underbrace{\boldsymbol{\delta}^{(1)} \leftarrow \cdots \leftarrow \boldsymbol{\delta}^{(L-1)} \leftarrow \boldsymbol{\delta}^{(L)}}_{\text{Back-prop}}$

Initialize $\boldsymbol{w}$?

- Init $\boldsymbol{w}$ with all zeros (or values equal to each other)?

  [blackboard]

- People often init $\boldsymbol{w}$ with small random values around 0 (“small” means “small absolute value”)

  [blackboard]