

Deep Learning for Natural Language Processing

Lecture 1: Logistics Regression for Text Classification

Quan Thanh Tho
Faculty of Computer Science and Engineering
Back Khoa University

Acknowledgement

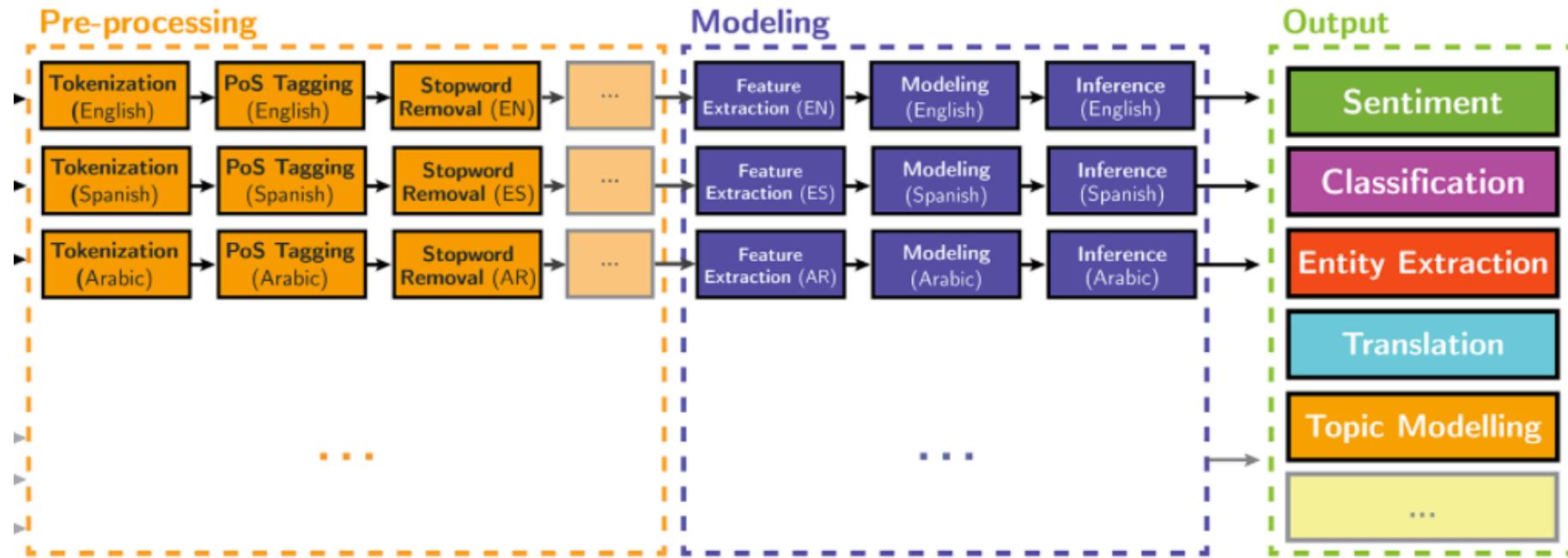
- Some slides are from Coursera course of Prof. Andrew Ng.

Agenda

- Classical Machine Learning Techniques for NLP
- Neural-based Roadmap of NLP
- Logistics Regression for Text Classification
- Demonstration and Assignment

Classical Machine Learning Techniques for NLP

- Machine Learning Task
- The tf.idf weights
- Neural-based approach





Document D = $D_1, D_2, D_3, \dots, D_m$

Word W = $W_1, W_2, W_3, \dots, W_m$

Make vector by tf-idf method

Vector space model

$D = [D_1, D_2, D_3, \dots, D_i, \dots, D_m] =$

Weight vector of
document "D_i"

$W = [W_1, W_2, W_3, \dots, W_j, \dots, W_n]$ Weight of word
W_i of

document D_i

α_{11}	α_{12}	...	α_{1j}	...	α_{1n}
α_{21}	α_{22}	...	α_{2j}	...	α_{2n}
:	:				:
α_{i1}	α_{i2}	...	α_{ij}	...	α_{in}
⋮	⋮				⋮
α_{m1}	α_{m2}	...	α_{mj}	...	α_{mn}

Document Collection

- A collection of n documents can be represented in the vector space model by a term-document matrix.
- An entry in the matrix corresponds to the “**weight**” of a term in the document; zero means the term has no significance in the document or it simply doesn’t exist in the document.

$$\begin{matrix} & T_1 & T_2 & \dots & T_t \\ \begin{pmatrix} D_1 & w_{11} & w_{21} & \dots & w_{t1} \\ D_2 & w_{12} & w_{22} & \dots & w_{t2} \\ \vdots & \vdots & \vdots & & \vdots \\ \vdots & \vdots & \vdots & & \vdots \\ D_n & w_{1n} & w_{2n} & \dots & w_{tn} \end{pmatrix} \end{matrix}$$

Term Weights: Term Frequency

- More frequent terms in a document are more important, i.e. more indicative of the topic.

$$f_{ij} = \text{frequency of term } i \text{ in document } j$$

- May want to normalize *term frequency* (*tf*) by dividing by the frequency of the most common term in the document:

$$tf_{ij} = f_{ij} / \max_i \{f_{ij}\}$$

Term Weights: Inverse Document Frequency

- Terms that appear in many *different* documents are *less* indicative of overall topic.

df_i = document frequency of term i

= number of documents containing term i

idf_i = inverse document frequency of term i ,

= $\log_2 (N / df_i)$

(N : total number of documents)

- An indication of a term's *discrimination power*.
- Log used to dampen the effect relative to tf .

TF-IDF Weighting

- A typical combined term importance indicator is *tf-idf weighting*:

$$w_{ij} = tf_{ij} \cdot idf_i = tf_{ij} \log_2 (N/df_i)$$

- A term occurring frequently in the document but rarely in the rest of the collection is given high weight.
- Many other ways of determining term weights have been proposed.
- Experimentally, *tf-idf* has been found to work well.

Computing TF-IDF -- An Example

Given a document containing terms with given frequencies:

A(3), B(2), C(1)

Assume collection contains 10,000 documents and
document frequencies of these terms are:

A(50), B(1300), C(250)

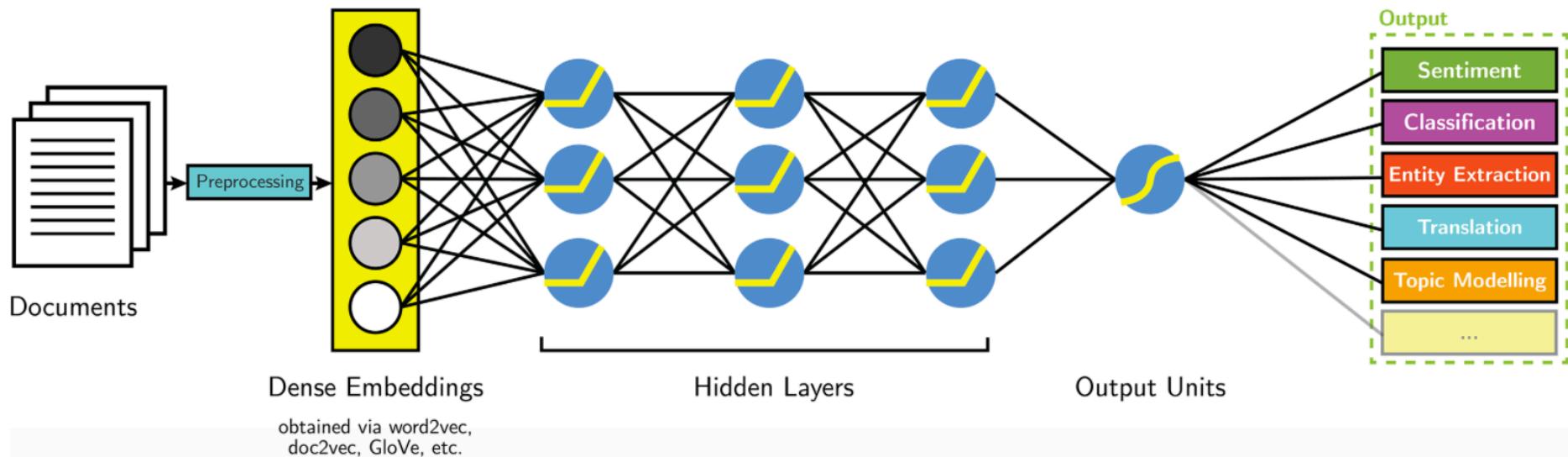
Then:

A: $\text{tf} = 3/3$; $\text{idf} = \log_2(10000/50) = 7.6$; $\text{tf-idf} = 7.6$

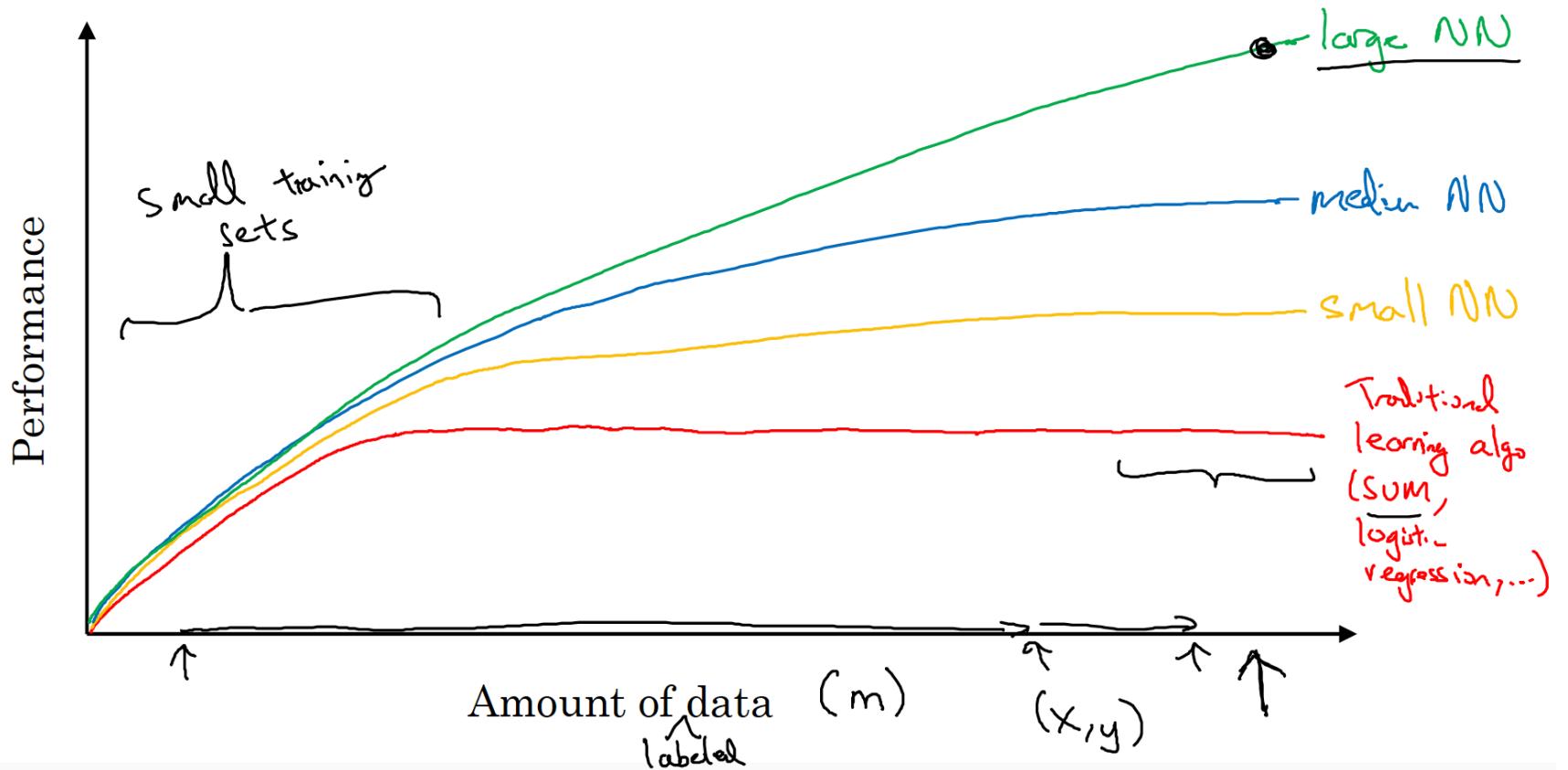
B: $\text{tf} = 2/3$; $\text{idf} = \log_2(10000/1300) = 2.9$; $\text{tf-idf} = 2.0$

C: $\text{tf} = 1/3$; $\text{idf} = \log_2(10000/250) = 5.3$; $\text{tf-idf} = 1.8$

Neural-based Approaches



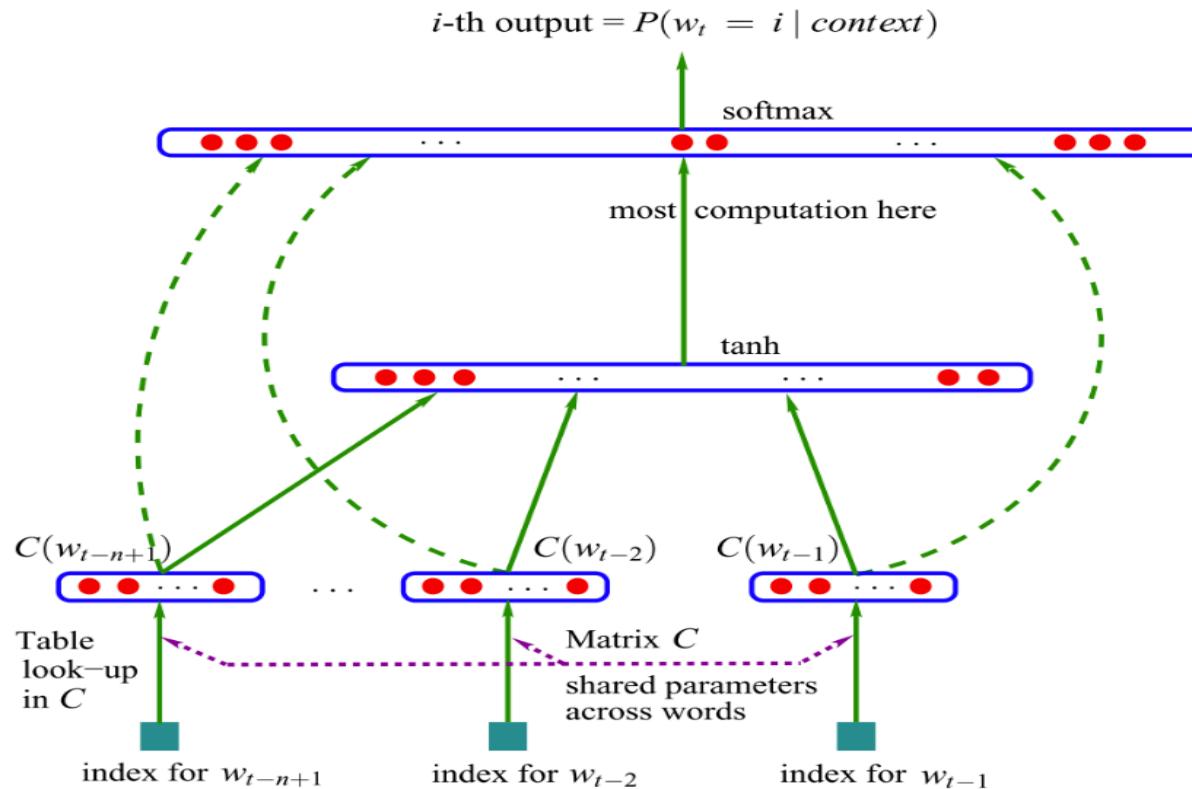
Why Neural?



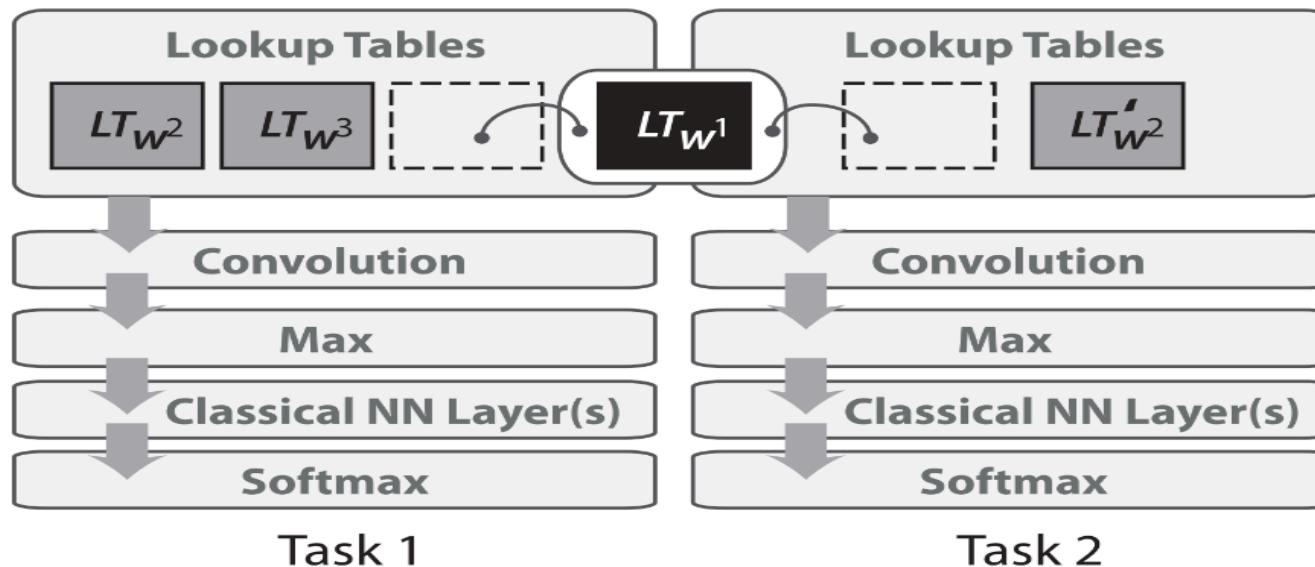
Neural-based Milestones for NLP

- 2001 - Neural language models
Character-based representations
Adversarial learning
Reinforcement learning
- 2008 - Multi-task learning
- 2013 - Word embeddings
- 2013 - Neural networks for NLP
- 2014 - Sequence-to-sequence models
Conditional random fields
Bleu
- 2015 - Memory-based networks
Latent Dirichlet Allocation
Wikipedia exploitation
- 2018 - Pretrained language models

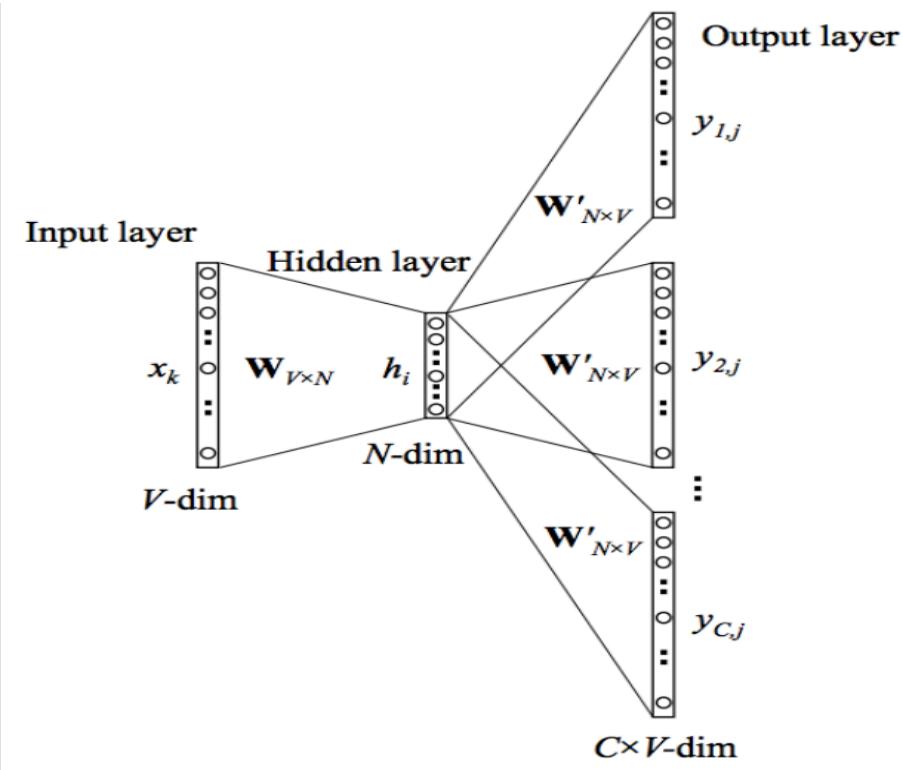
Neural Language Model



Multitask Learning



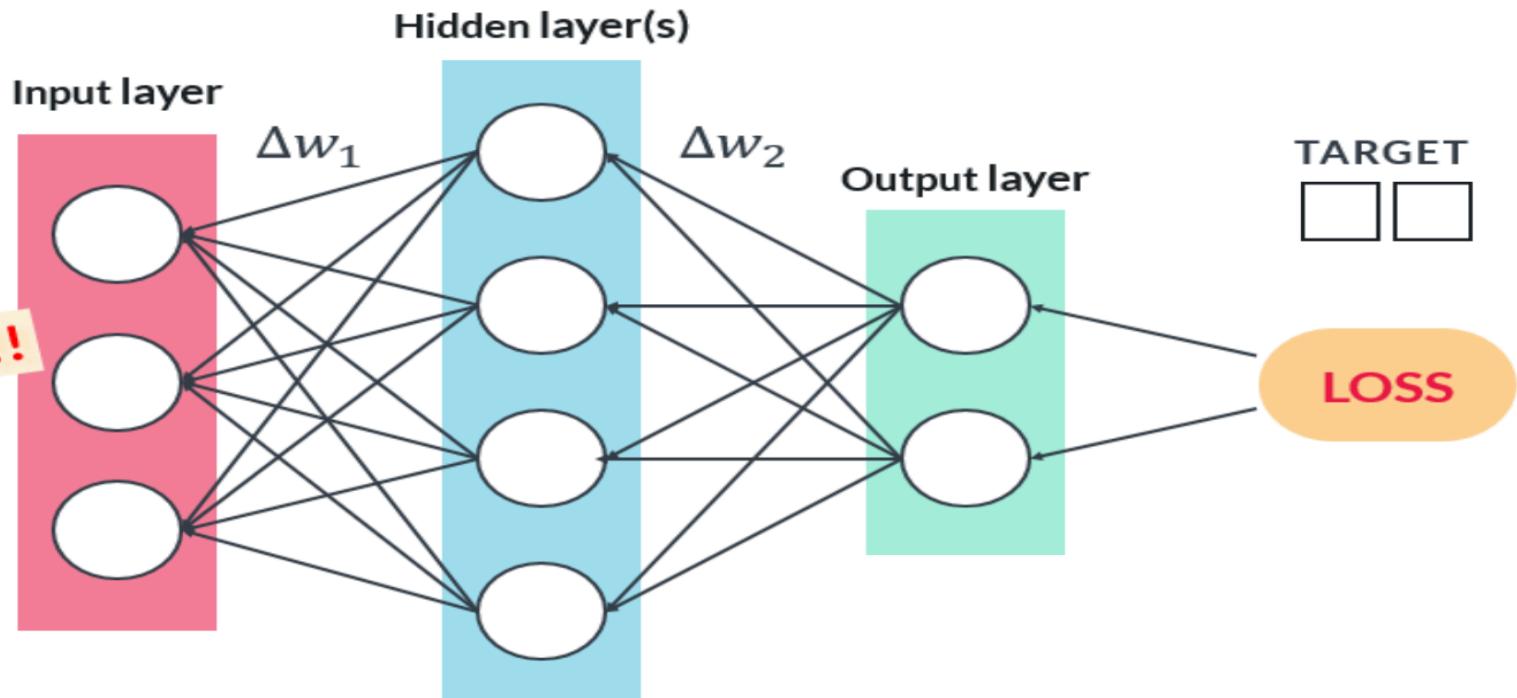
Word Embedding



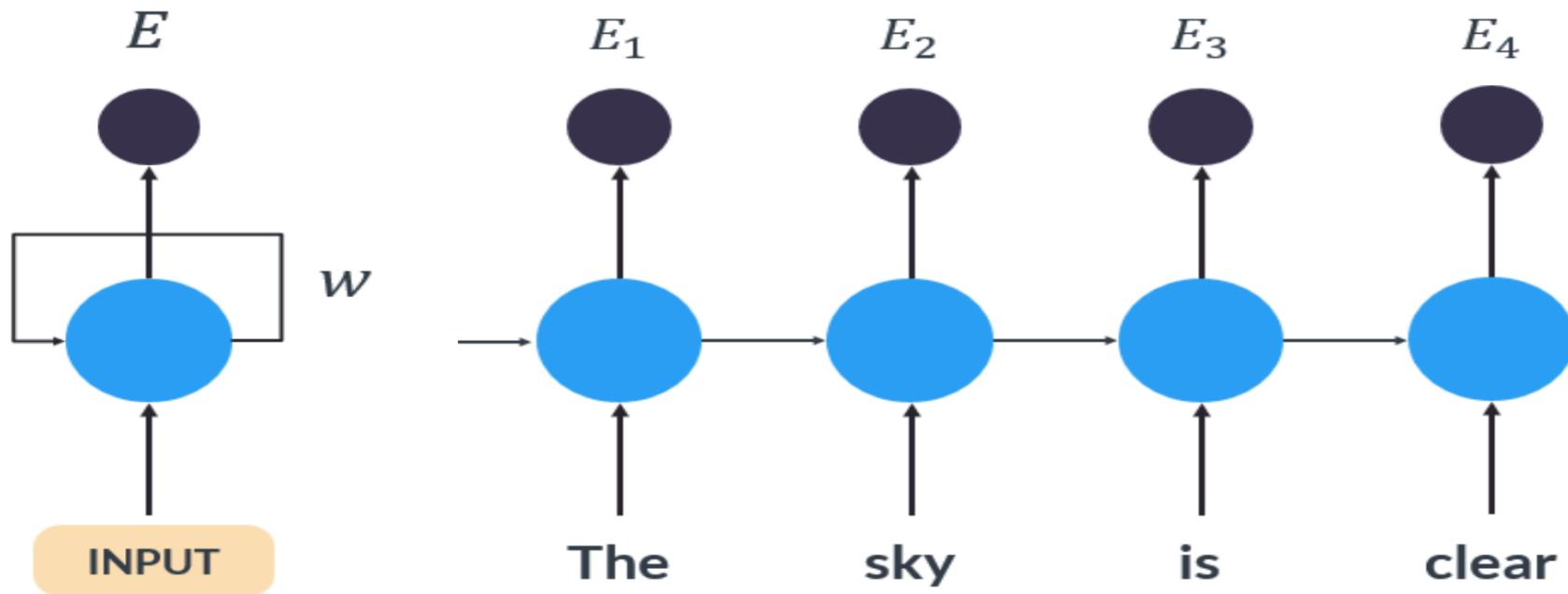
FIXED VECTOR



PROBLEM!!

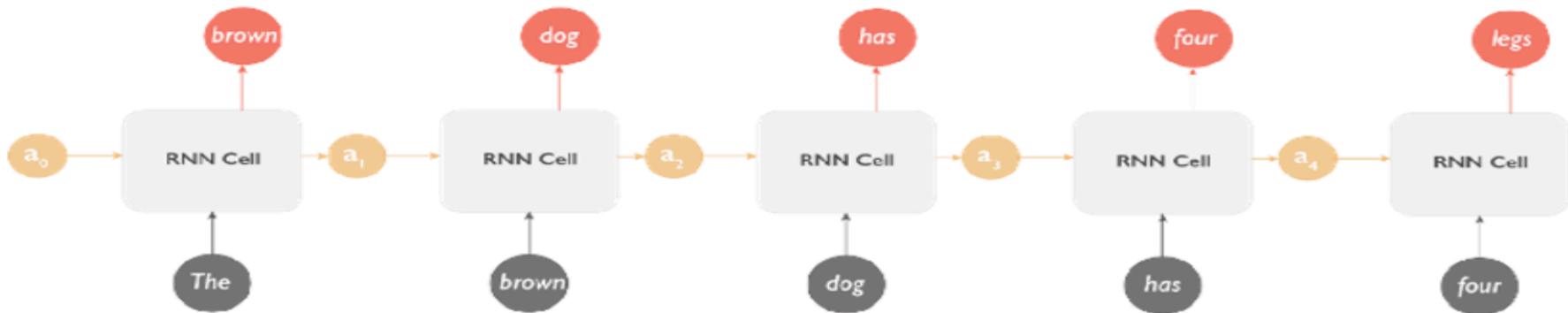


Recurrent Neural Networks



RNN Common Architectures

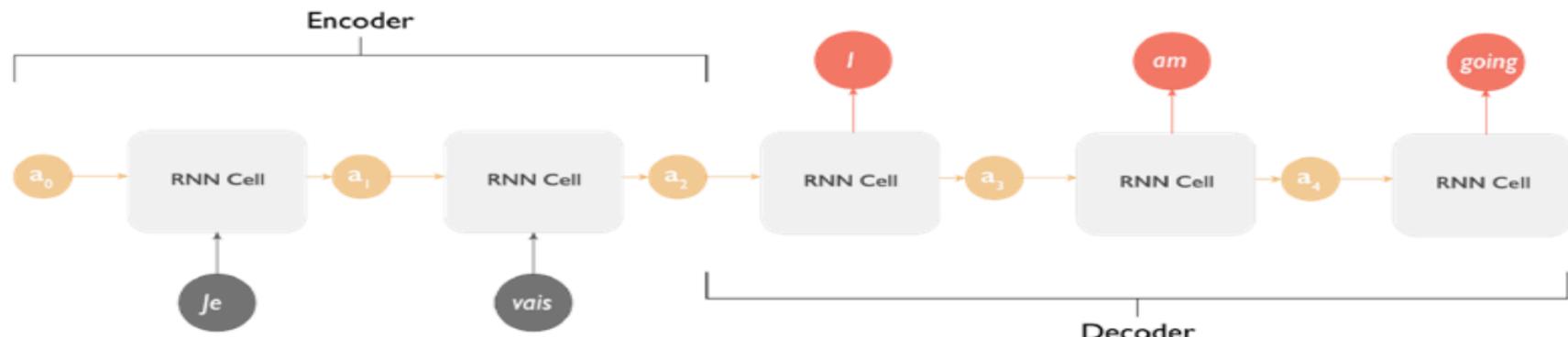
- **Many-to-many (same sequence lengths)**



- **Many-to-many (different sequence lengths)**
- **Many-to-one**
- **One-to-many**

RNN Common Architectures

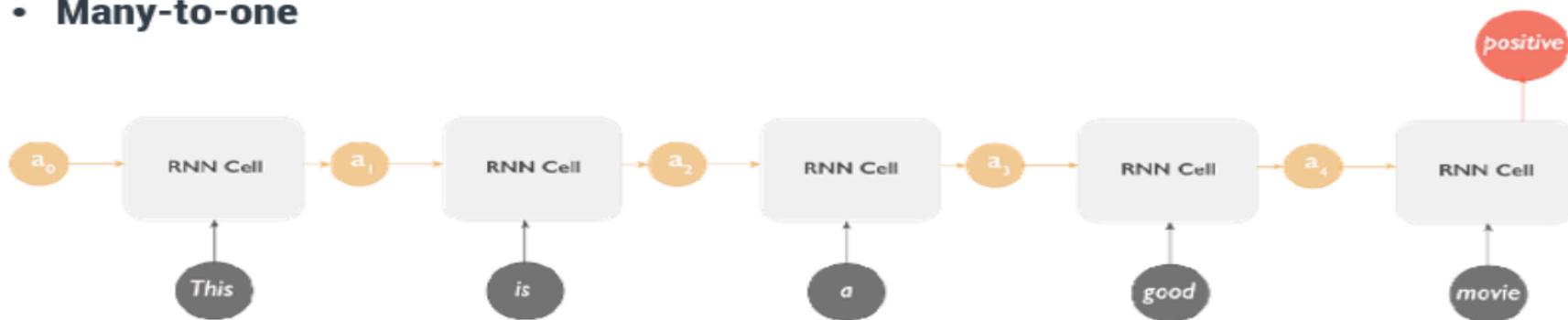
- Many-to-many (same sequence lengths)
- Many-to-many (different sequence lengths)



- Many-to-one
- One-to-many

RNN Common Architectures

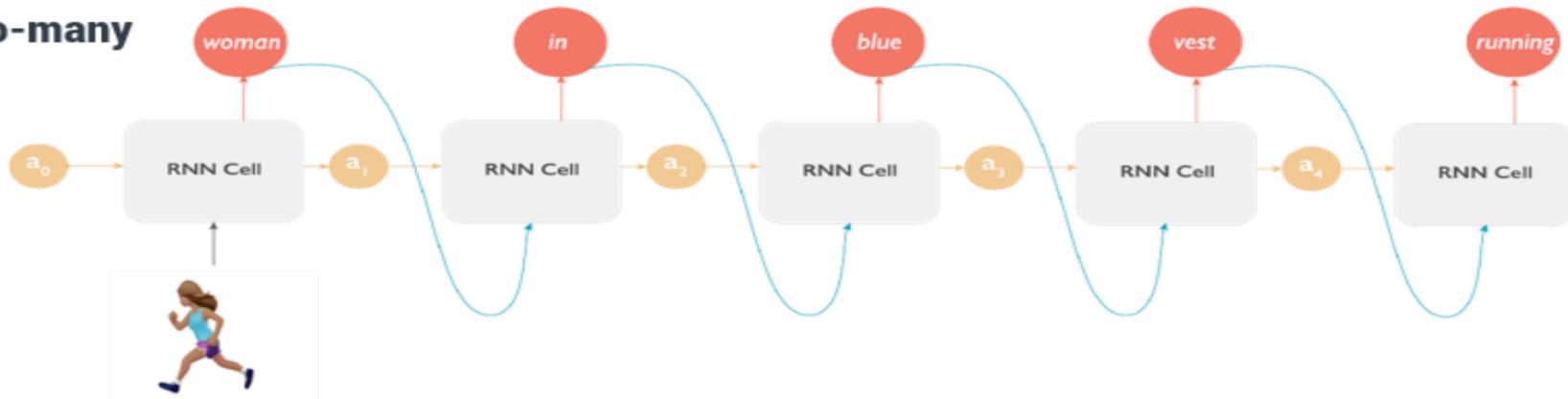
- Many-to-many (same sequence lengths)
- Many-to-many (different sequence lengths)
- Many-to-one



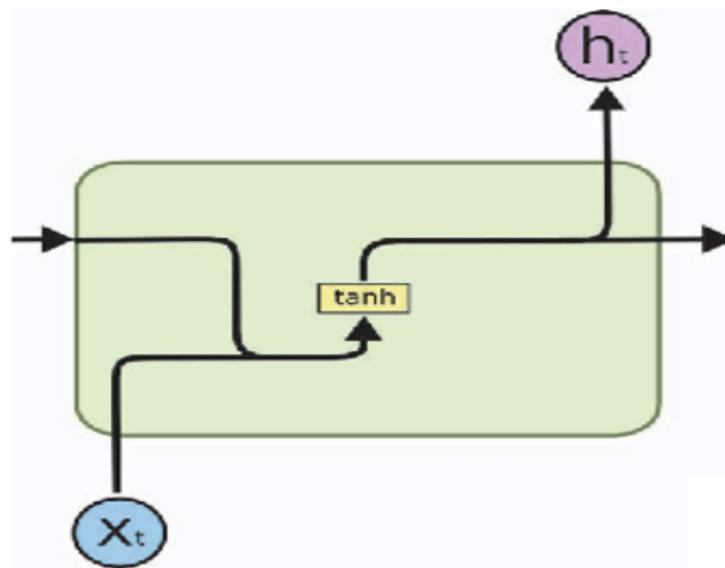
- One-to-many

RNN Common Architectures

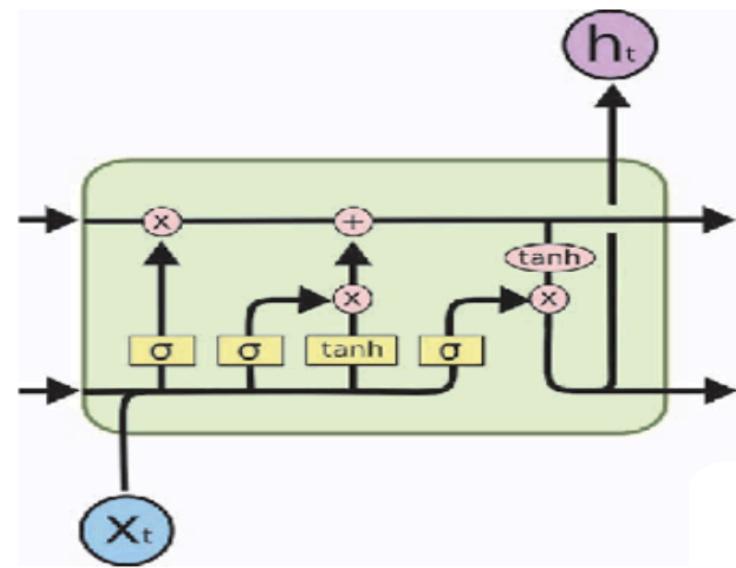
- **Many-to-many (same sequence lengths)**
- **Many-to-many (different sequence lengths)**
- **Many-to-one**
- **One-to-many**



Enhancement from RNN

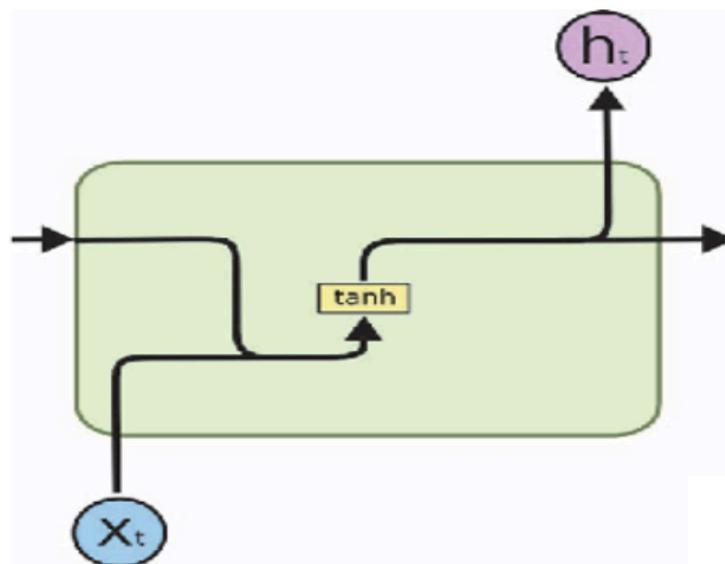


RNN

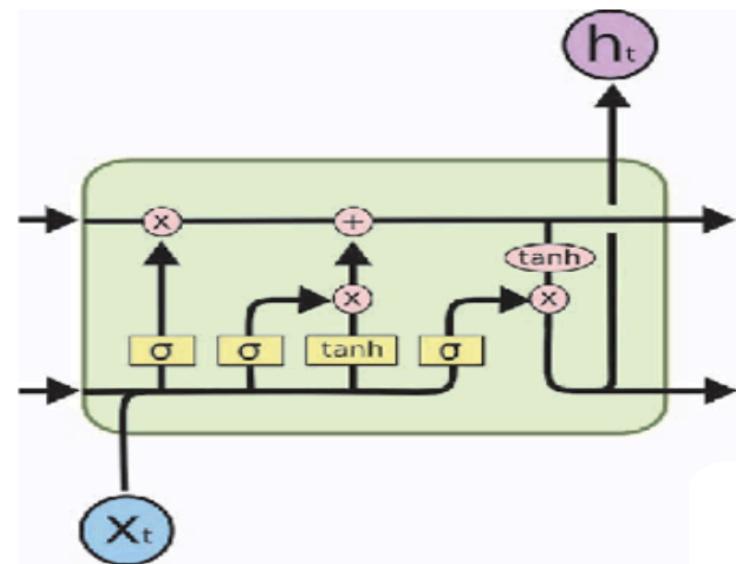


LSTM

Enhancement from RNN

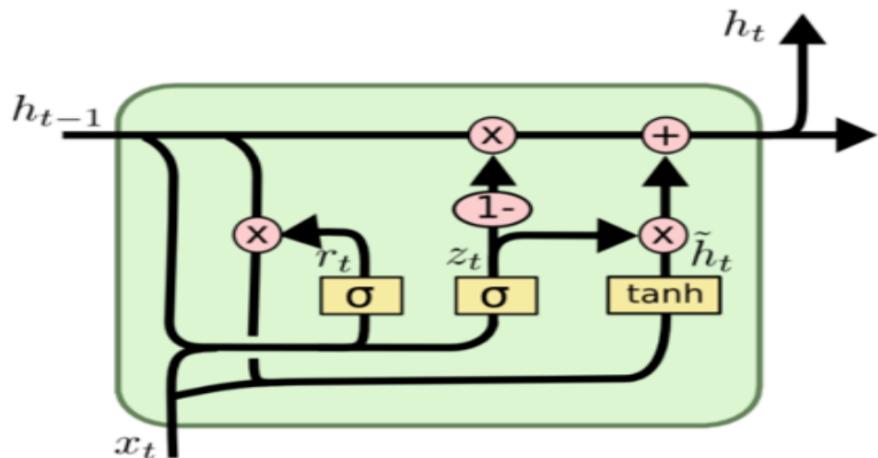


RNN



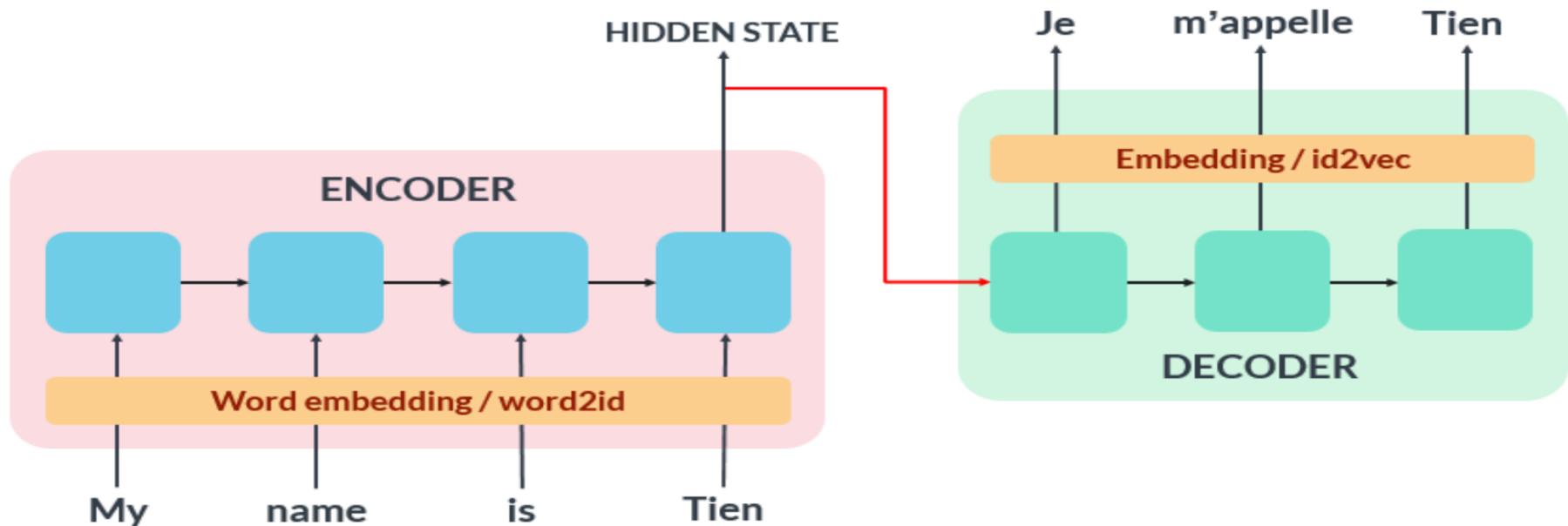
LSTM

Gated Recurrent Unit

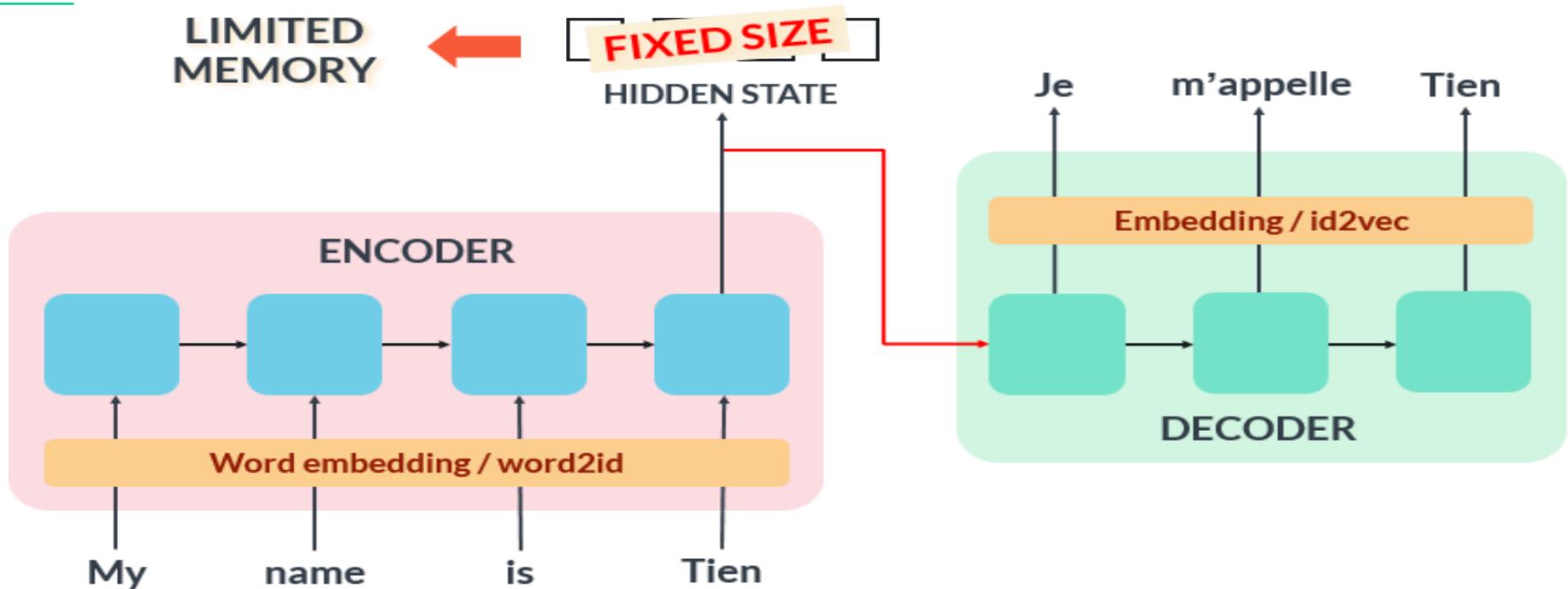


- Similar to LSTM but with simpler structure.
- Combine **forget** & **input gate** into “**update gate**”.
- Also combine **output gate** & **hidden state**.

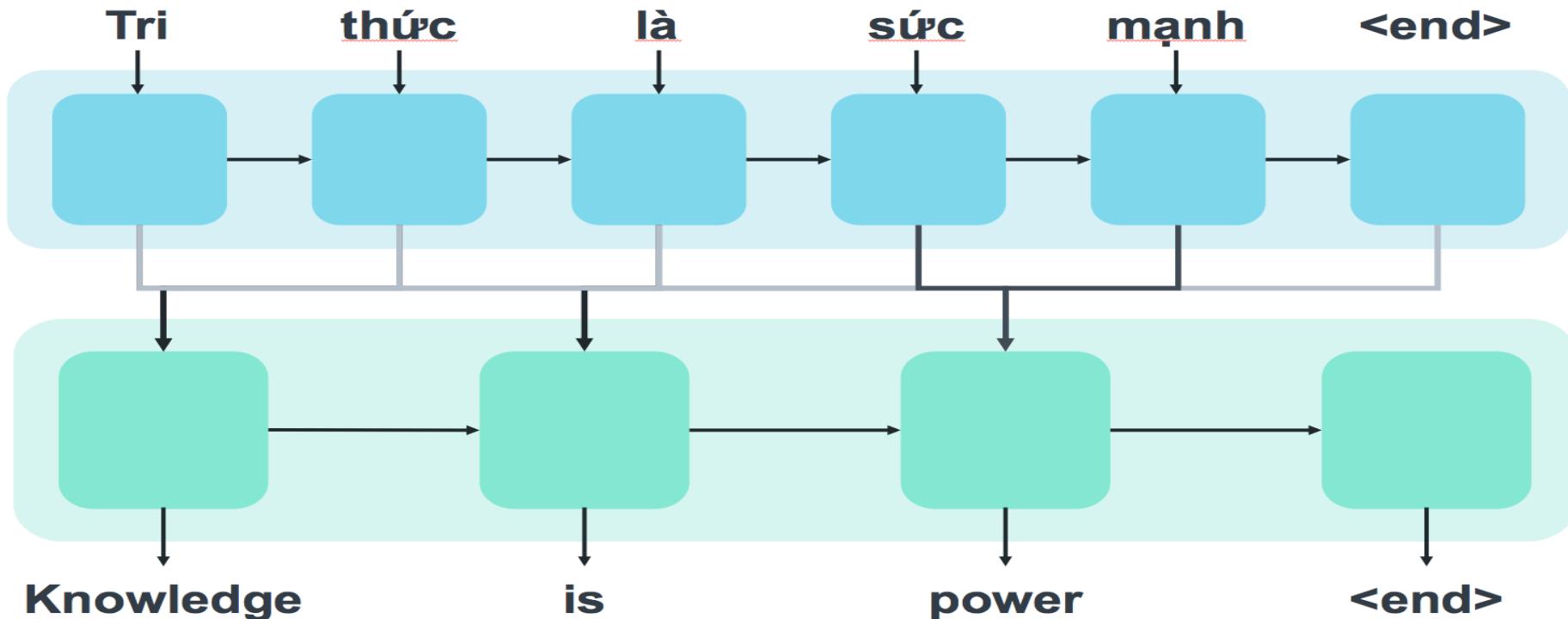
Seq2seq Architecture



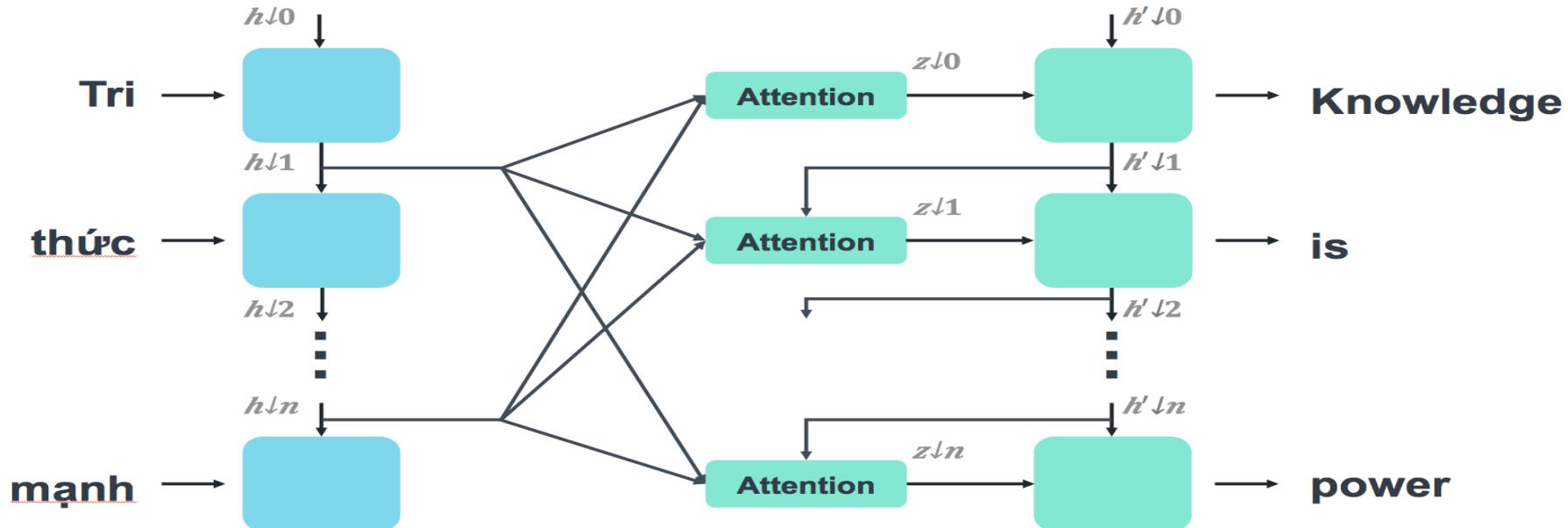
Seq2seq limitation



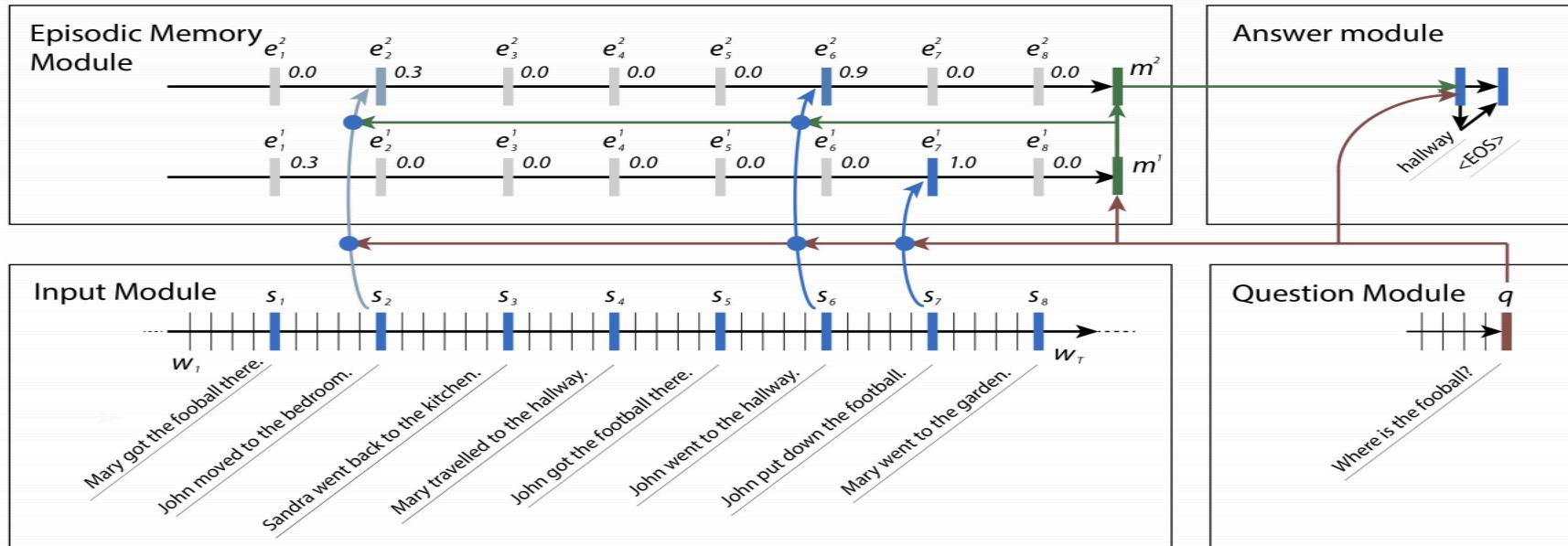
Attention Mechanism



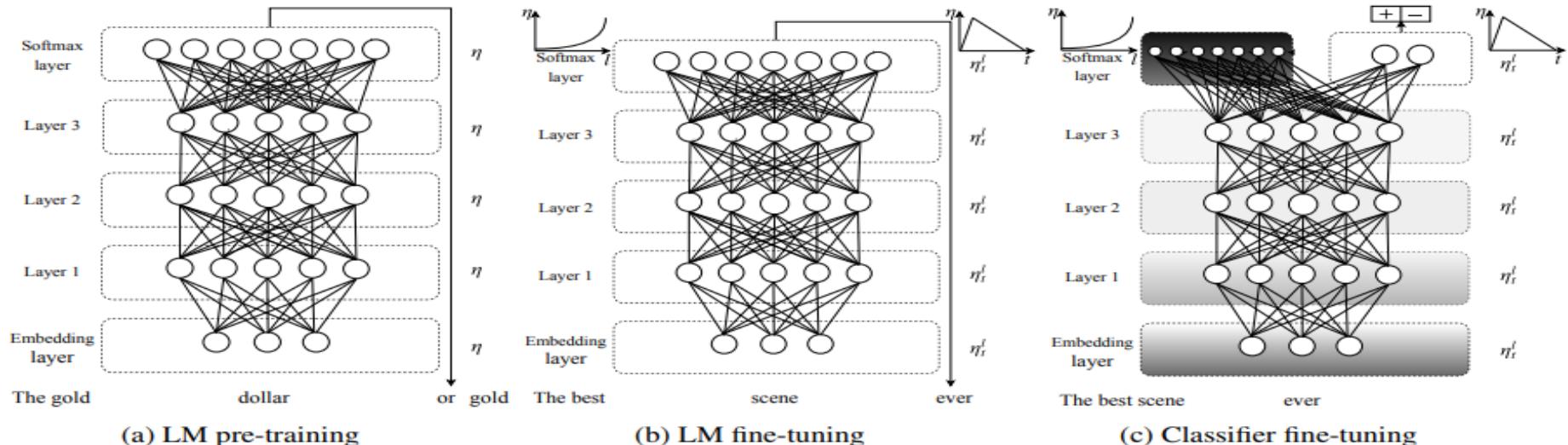
Attention Mechanism



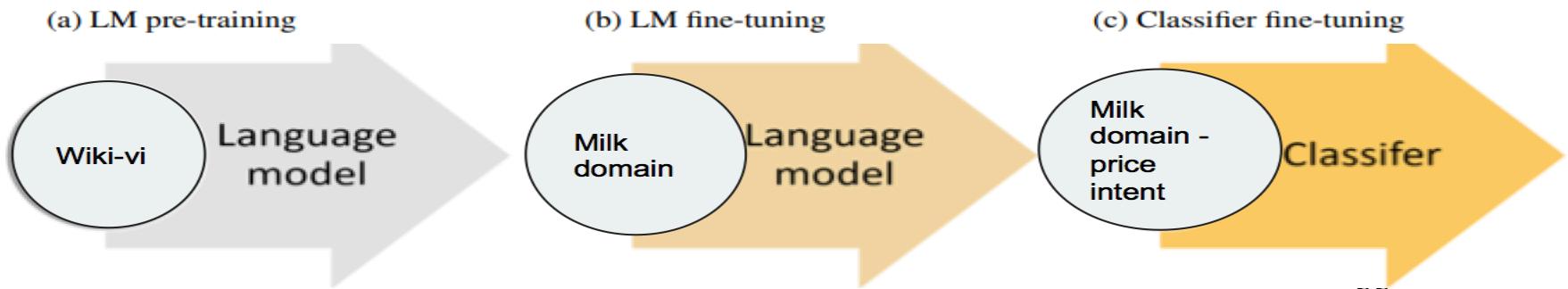
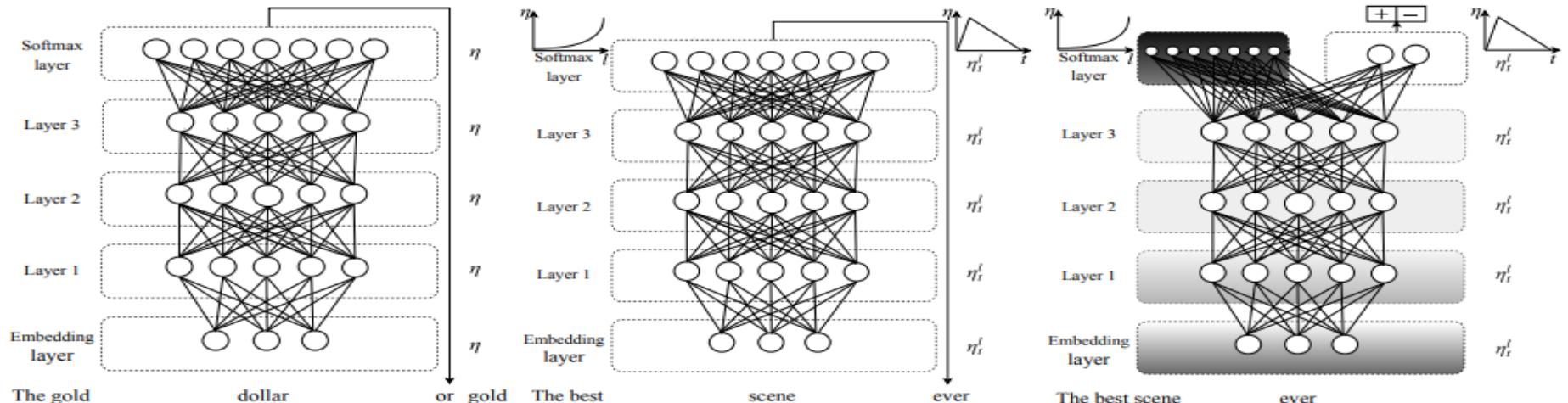
Dynamic Memory Model



Pre-trained Language Model



Pre-trained Neural Language Model



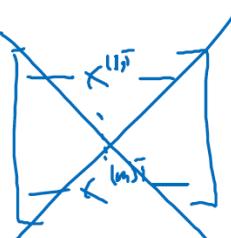
Supervised Learning Problem

(x, y) $x \in \mathbb{R}^{n_x}$, $y \in \{0, 1\}$

m training examples : $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})\}$

$M = M_{\text{train}}$

$M_{\text{test}} = \# \text{test examples.}$

$$X = \begin{bmatrix} | & | & | \\ x^{(1)} & x^{(2)} & \dots & x^{(m)} \\ | & | & | \end{bmatrix} \quad \begin{array}{c} \uparrow \\ n_x \\ \downarrow \end{array}$$


$X \in \mathbb{R}^{n_x \times m}$

$X.\text{shape} = (n_x, m)$

$$Y = [y^{(1)} \ y^{(2)} \ \dots \ y^{(m)}]$$

$Y \in \mathbb{R}^{1 \times m}$

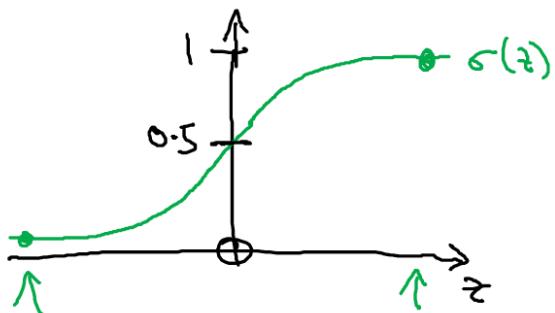
$Y.\text{shape} = (1, m)$

Logistic Regression

Given x , want $\hat{y} = \frac{P(y=1|x)}{0 \leq \hat{y} \leq 1}$
 $x \in \mathbb{R}^{n_x}$

Parameters: $w \in \mathbb{R}^{n_x}$, $b \in \mathbb{R}$.

Output $\hat{y} = \sigma(w^T x + b)$



$$x_0 = 1, \quad x \in \mathbb{R}^{n_x+1}$$

$$\hat{y} = \sigma(w^T x)$$

$$w = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_{n_x} \end{bmatrix} \quad \left\{ \begin{array}{l} b \leftarrow w_0 \\ w \leftarrow \{w_1, w_2, \dots, w_{n_x}\} \end{array} \right.$$

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

$$\text{If } z \text{ large } \sigma(z) \approx \frac{1}{1+0} = 1$$

If z large negative number

$$\sigma(z) = \frac{1}{1 + e^{-z}} \approx \frac{1}{1 + \text{Bignum}} \approx 0$$

Logistic Regression cost function

$$\rightarrow \hat{y}^{(i)} = \sigma(w^T x^{(i)} + b), \text{ where } \sigma(z^{(i)}) = \frac{1}{1+e^{-z^{(i)}}} \quad z^{(i)} = w^T x^{(i)} + b$$

Given $\{(x^{(1)}, y^{(1)}), \dots, (x^{(m)}, y^{(m)})\}$, want $\hat{y}^{(i)} \approx y^{(i)}$.

$\begin{matrix} x^{(i)} \\ y^{(i)} \\ z^{(i)} \end{matrix}$
-th example.

Loss (error) function: $L(\hat{y}, y) = \frac{1}{2} (\hat{y} - y)^2$



$$L(\hat{y}, y) = - (y \log \hat{y} + (1-y) \log(1-\hat{y})) \leftarrow$$

If $y=1$: $L(\hat{y}, y) = - \log \hat{y} \leftarrow$ want $\log \hat{y}$ large, want \hat{y} large.

If $y=0$: $L(\hat{y}, y) = - \log(1-\hat{y}) \leftarrow$ want $\log(1-\hat{y})$ large ... want \hat{y} small

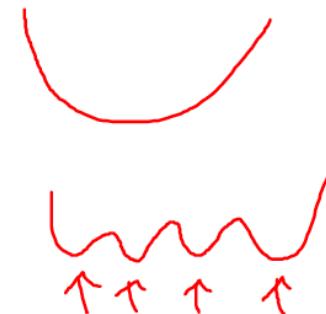
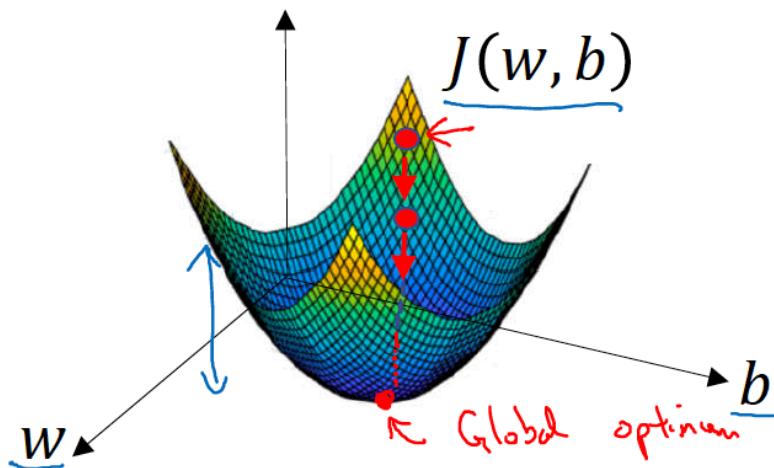
Cost function: $J(w, b) = \frac{1}{m} \sum_{i=1}^m L(\hat{y}^{(i)}, y^{(i)}) = \frac{1}{m} \sum_{i=1}^m [y^{(i)} \log \hat{y}^{(i)} + (1-y^{(i)}) \log(1-\hat{y}^{(i)})]$

Gradient Descent

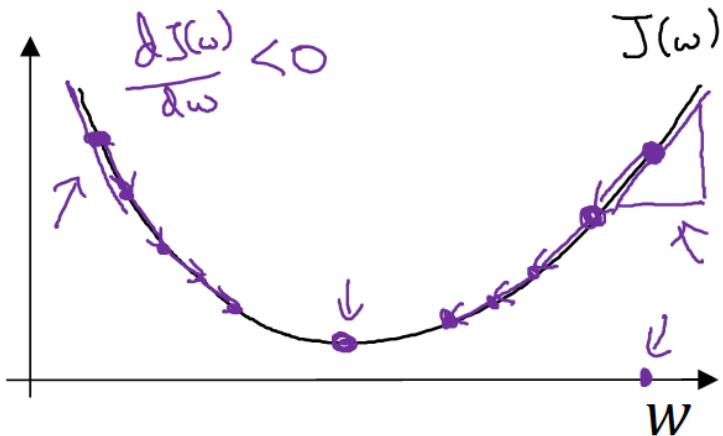
Recap: $\hat{y} = \sigma(w^T x + b)$, $\sigma(z) = \frac{1}{1+e^{-z}}$ ←

$$J(w, b) = \frac{1}{m} \sum_{i=1}^m \mathcal{L}(\hat{y}^{(i)}, y^{(i)}) = -\frac{1}{m} \sum_{i=1}^m y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})$$

Want to find w, b that minimize $J(w, b)$



Gradient Descent



Repeat {
 $w := w - \alpha$
 $w := w - \alpha \frac{dJ(w)}{dw}$
 ↑ ↑
 "dw"
 }
 $\frac{dJ(w)}{dw} = ?$

$$J(w, b)$$

$$w := w - \alpha \frac{dJ(w, b)}{dw}$$

$$b := b - \alpha \frac{dJ(w, b)}{db}$$

"partial derivative" J

$\frac{\partial J(w, b)}{\partial w}$ $\frac{\partial J(w, b)}{\partial b}$

d d

dw db

Computation Graph

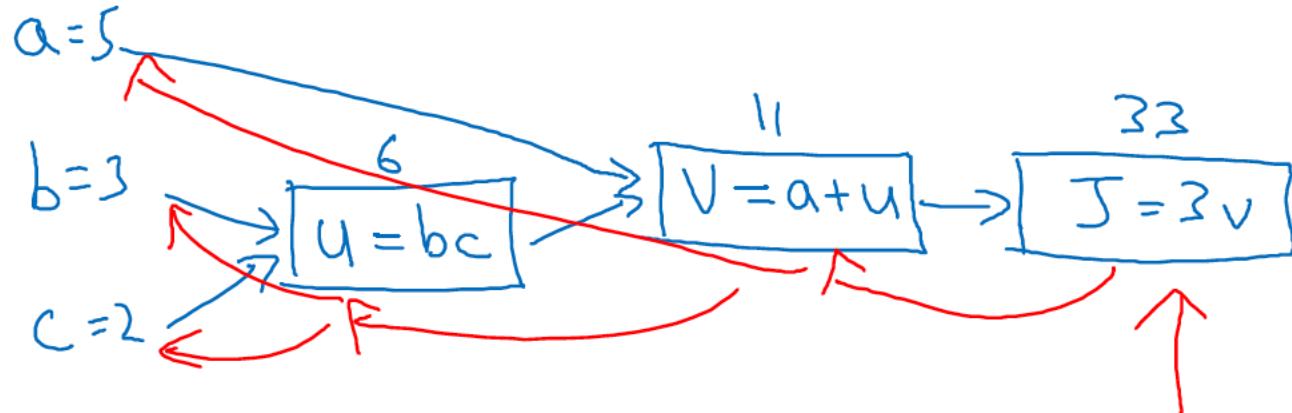
$$J(a, b, c) = 3(u + bc) = 3(5 + 3 \cdot 2) = 33$$

\underbrace{u}_{v}
 \underbrace{v}_{J}

$$u = bc$$

$$v = a + u$$

$$J = 3v$$



Computing derivatives

$$a = 5 \quad \frac{dJ}{da} \text{ "d}a\text{"} = 3$$

$$b = 3$$

$$c = 2$$

$$u = bc \quad 6$$

(11)

$$v = a + u$$

$$J = 3v$$

$$\frac{dJ}{dv} \text{ "d}v\text{"} = 3$$

$$\frac{dJ}{dv} = ? = 3$$

$a \rightarrow v \rightarrow J$

$$J = 3v$$

$$v = 11 \rightarrow 11.001$$

$$J = 33 \rightarrow 33.003$$

$$\frac{dJ}{da} = 3 = \frac{dJ}{dv} \frac{dv}{da}$$

$\cancel{3} \times \cancel{1}$

$$\frac{dv}{da} = 1$$

$$\frac{\partial \text{Final Output Var}}{\partial \text{var}}$$

$$\frac{\partial J}{\partial \text{var}}$$

"dvar"

$$f(a) = 3a$$

$$\frac{df(b)}{da} = \frac{df}{da} = 3$$

$$J = 3v$$

$$\frac{dJ}{dv} = 3$$

Computing derivatives

Given:

$$\begin{aligned} \frac{\partial J}{\partial a} &= 5 \\ \frac{\partial J}{\partial b} &= 3 \\ \frac{\partial J}{\partial c} &= 2 \end{aligned}$$

Intermediate variables:

$$v = a + u \quad J = 3v$$
$$u = bc \quad \frac{\partial v}{\partial u} = 3 \quad \frac{\partial J}{\partial v} = 3$$

Final result:

$$\frac{\partial J}{\partial u} = 3 = \frac{\partial J}{\partial v} \cdot \frac{\partial v}{\partial u}$$

Derivatives with respect to b and c :

$$\frac{\partial J}{\partial b} = \frac{\partial J}{\partial u} \cdot \frac{\partial u}{\partial b} = 6$$
$$\frac{\partial J}{\partial c} = \frac{\partial J}{\partial u} \cdot \frac{\partial u}{\partial c} = 9$$

Approximate values:

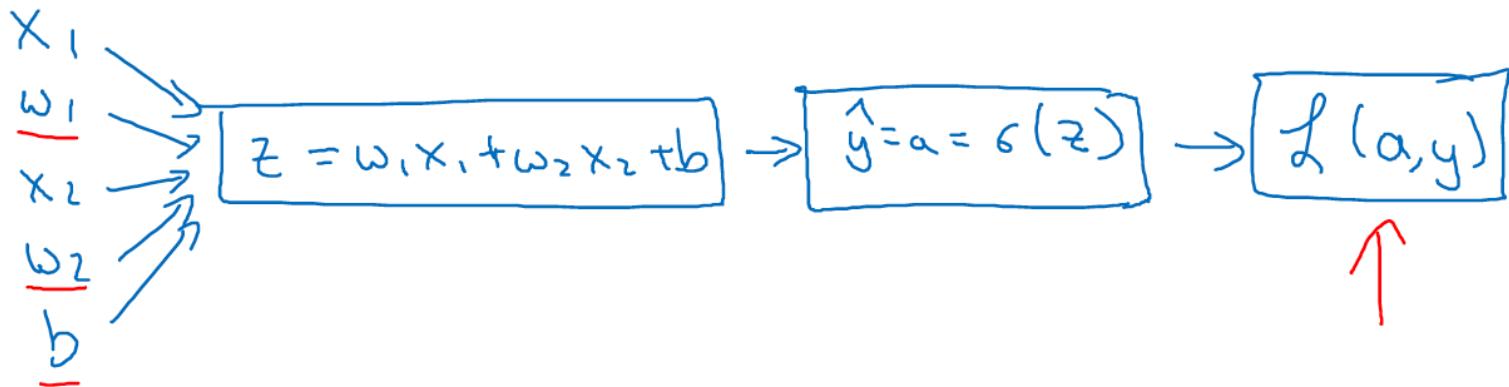
$$\begin{aligned} u &= 6 \rightarrow 6.001 \\ v &= 11 \rightarrow 11.001 \\ J &= 33 \rightarrow 33.003 \end{aligned}$$
$$\begin{aligned} b &= 3 \rightarrow 3.001 \\ u &= b \cdot c = 6 \rightarrow 6.002 \\ J &= 33.006 \end{aligned}$$
$$\begin{aligned} c &= 2 \\ &\quad \cdot 006 \\ v &= 11.002 \\ J &= 33.008 \end{aligned}$$

Logistic regression recap

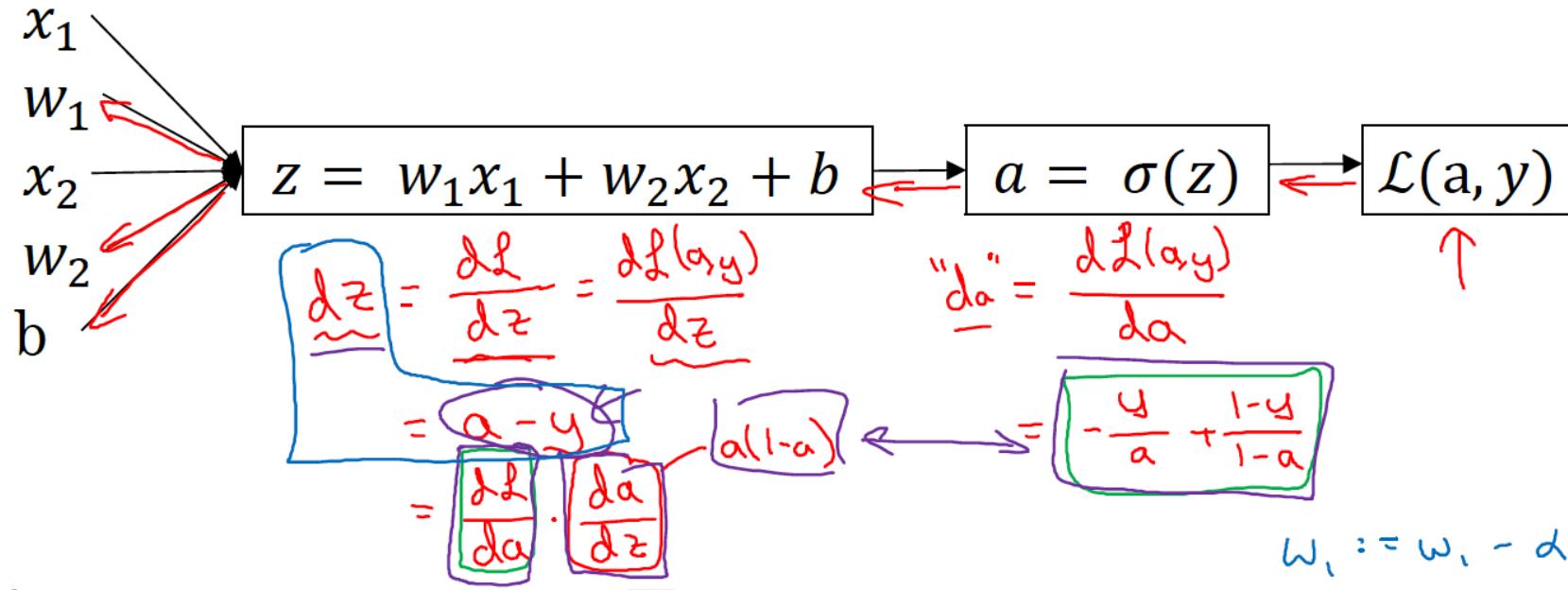
$$\rightarrow z = w^T x + b$$

$$\rightarrow \hat{y} = a = \sigma(z)$$

$$\rightarrow \mathcal{L}(a, y) = -(y \log(a) + (1 - y) \log(1 - a))$$



Logistic regression derivatives



$$\frac{\partial \mathcal{L}}{\partial w_1} = "d_{w_1}" = x_1 \cdot dz.$$

$$d_{w_1} = x_1 \cdot dz. \quad db = dz.$$

$$w_1 := w_1 - \alpha d_{w_1}$$

$$w_2 := w_2 - \alpha d_{w_2}$$

$$b := b - \alpha db.$$

Logistic regression on m examples

$$\underline{J(\omega, b)} = \frac{1}{m} \sum_{i=1}^m \underline{\ell(a^{(i)}, y^{(i)})}$$
$$\rightarrow a^{(i)} = \hat{y}^{(i)} = g(z^{(i)}) = g(\omega^\top x^{(i)} + b)$$
$$\underline{d\omega_1^{(i)}}, \underline{d\omega_2^{(i)}}, \underline{db^{(i)}}$$
$$(x^{(i)}, y^{(i)})$$

$$\underline{\frac{\partial}{\partial \omega_1} J(\omega, b)} = \frac{1}{m} \sum_{i=1}^m \underbrace{\frac{\partial}{\partial \omega_1} \ell(a^{(i)}, y^{(i)})}_{d\omega_1^{(i)}} - (x^{(i)}, y^{(i)})$$

Logistic regression on m examples

$$J=0; \underline{dw_1}=0; \underline{dw_2}=0; \underline{db}=0$$

→ For $i = 1$ to m

$$z^{(i)} = \omega^\top x^{(i)} + b$$

$$\alpha^{(i)} = \sigma(z^{(i)})$$

$$J_t = -[y^{(i)} \log \alpha^{(i)} + (1-y^{(i)}) \log (1-\alpha^{(i)})]$$

$$\underline{dz^{(i)}} = \alpha^{(i)} - y^{(i)}$$

$$\begin{aligned} dw_1 &+ x_1^{(i)} dz^{(i)} \\ dw_2 &+ x_2^{(i)} dz^{(i)} \\ db &+ dz^{(i)} \end{aligned}$$

$$J/m \leftarrow$$

$$dw_1/m; dw_2/m; db/m \leftarrow$$

$$dw_1 = \frac{\partial J}{\partial w_1}$$

$$w_1 := w_1 - \alpha \underline{dw_1}$$

$$w_2 := w_2 - \alpha \underline{dw_2}$$

$$b := b - \alpha \underline{db}.$$

Vectorization

Vectorizing Logistic Regression

$$\begin{aligned} \rightarrow z^{(1)} &= w^T x^{(1)} + b \\ \rightarrow a^{(1)} &= \sigma(z^{(1)}) \end{aligned}$$

$$\begin{aligned} z^{(2)} &= w^T x^{(2)} + b \\ a^{(2)} &= \sigma(z^{(2)}) \end{aligned}$$

$$\begin{aligned} z^{(3)} &= w^T x^{(3)} + b \\ a^{(3)} &= \sigma(z^{(3)}) \end{aligned}$$

$$\underline{\underline{X}} = \begin{bmatrix} x^{(1)} & x^{(2)} & \dots & x^{(m)} \\ | & | & \dots & | \\ 1 & 1 & \dots & 1 \end{bmatrix}$$

$$\frac{(n_x, m)}{R^{n_x \times m}}$$

$$\stackrel{\text{---}}{\omega} \rightarrow \begin{bmatrix} 1 & 1 & \dots & 1 \\ x^{(1)} & x^{(2)} & \dots & x^{(m)} \\ | & | & \dots & | \end{bmatrix}$$

$$\begin{aligned} \underline{\underline{Z}} &= \begin{bmatrix} z^{(1)} & z^{(2)} & \dots & z^{(m)} \end{bmatrix} = \underline{\underline{w}^T X} + \begin{bmatrix} b & b & \dots & b \end{bmatrix}_{1 \times m} = \begin{bmatrix} w^T x^{(1)} + b \\ \vdots \\ w^T x^{(m)} + b \end{bmatrix} \dots \begin{bmatrix} w^T x^{(1)} + b \\ \vdots \\ w^T x^{(m)} + b \end{bmatrix}_{1 \times m} \\ \rightarrow \underline{\underline{Z}} &= np.\text{dot}(\underline{\underline{w}}, \underline{\underline{X}}) + \underline{\underline{b}}_{(1, 1)} \quad R \end{aligned}$$

"Broadcasting"

$$\underline{\underline{A}} = \begin{bmatrix} a^{(1)} & a^{(2)} & \dots & a^{(m)} \end{bmatrix} = \underline{\underline{\sigma(Z)}}$$

Vectorizing Logistic Regression

$$d_z^{(1)} = a^{(1)} - y^{(1)} \quad d_z^{(2)} = a^{(2)} - y^{(2)} \quad \dots$$

$d\bar{z} = [d_z^{(1)} \ d_z^{(2)} \ \dots \ d_z^{(m)}]_{1 \times m}$

$$A = [a^{(1)} \ \dots \ a^{(m)}]. \quad Y = [y^{(1)} \ \dots \ y^{(m)}]$$

$$\rightarrow d\bar{z} = A - Y = [a^{(1)} - y^{(1)} \ a^{(2)} - y^{(2)} \ \dots]$$

$$\begin{aligned} \rightarrow dw &= 0 \\ dw + &= \frac{x^{(1)} d\bar{z}^{(1)}}{} \\ dw + &= \frac{x^{(2)} d\bar{z}^{(2)}}{} \\ &\vdots \\ dw &= m \end{aligned}$$

$$\begin{aligned} db &= 0 \\ db + &= d\bar{z}^{(1)} \\ db + &= d\bar{z}^{(2)} \\ &\vdots \\ db + &= d\bar{z}^{(m)} \\ db &= m. \end{aligned}$$

$$\begin{aligned} db &= \frac{1}{m} \sum_{i=1}^m d\bar{z}^{(i)} \\ &= \frac{1}{m} \text{np. sum}(d\bar{z}) \end{aligned}$$

$$\begin{aligned} dw &= \frac{1}{m} \times d\bar{z}^\top \\ &= \frac{1}{m} \left[\underbrace{\begin{matrix} x^{(1)} & \dots & x^{(m)} \\ | & & | \end{matrix}}_{n \times 1} \right] \left[\begin{matrix} d\bar{z}^{(1)} \\ \vdots \\ d\bar{z}^{(m)} \end{matrix} \right] \\ &= \frac{1}{m} \left[\underbrace{\frac{x^{(1)} d\bar{z}^{(1)}}{}}_{n \times 1} + \dots + \underbrace{\frac{x^{(m)} d\bar{z}^{(m)}}{}}_{n \times 1} \right] \end{aligned}$$

Implementing Logistic Regression

$$J = 0, dw_1 = 0, dw_2 = 0, db = 0$$

for i = 1 to m:

$$z^{(i)} = w^T x^{(i)} + b \leftarrow$$

$$a^{(i)} = \sigma(z^{(i)}) \leftarrow$$

$$J += -[y^{(i)} \log a^{(i)} + (1 - y^{(i)}) \log(1 - a^{(i)})]$$
$$dz^{(i)} = a^{(i)} - y^{(i)} \leftarrow$$

$$\left. \begin{array}{l} dw_1 += x_1^{(i)} dz^{(i)} \\ dw_2 += x_2^{(i)} dz^{(i)} \end{array} \right\} \partial w + x^{(:)} * dz^{(i)}$$

$$db += dz^{(i)}$$

$$J = J/m, dw_1 = dw_1/m, dw_2 = dw_2/m$$

$$db = db/m$$

```
for iter in range(1000):  
    z = w^T X + b  
    = np.dot(w.T, X) + b  
    A = s(z)  
    dZ = A - Y  
    dw = 1/m * X * dZ^T  
    db = 1/m * np.sum(dZ)  
  
    w := w - alpha * dw  
    b := b - alpha * db
```