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

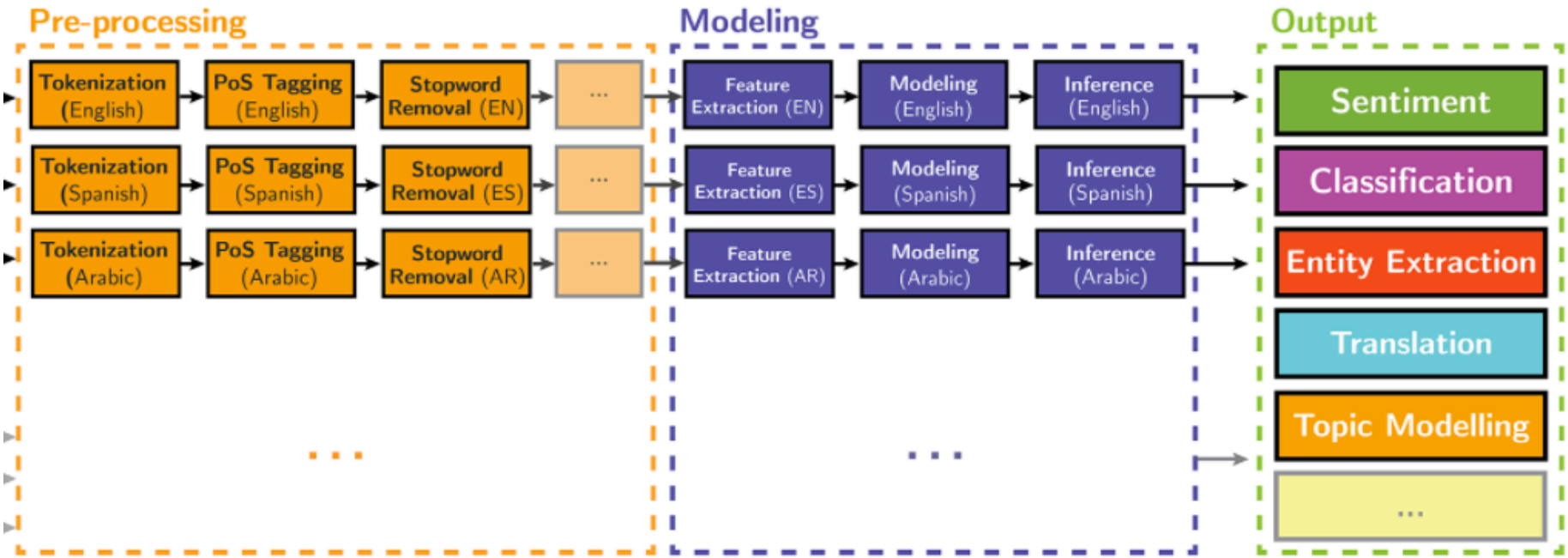
- Some slides are from Coursera course of Prof. Andew Ng.

Agenda

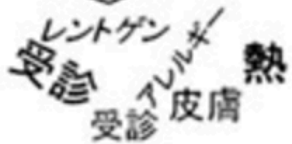
- Classical Machine Learning Techniques for NLP
- Neural-based Roadmap of NLP
- Logistics Regression for Text Classification
- Demonstration and Assignment

Classical Machine Learning Techniques for NLP

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


Pre-processing
Tokenization (English)
PoS Tagging (English)
Stopword Removal (EN)
...
Tokenization (Spanish)
PoS Tagging (Spanish)
Stopword Removal (ES)
...
Tokenization (Arabic)
PoS Tagging (Arabic)
Stopword Removal (AR)
...
Modeling
Feature Extraction (EN)
Modeling (English)
Inference (English)
Feature Extraction (ES)
Modeling (Spanish)
Inference (Spanish)
Feature Extraction (AR)
Modeling (Arabic)
Inference (Arabic)
Output
Sentiment
Classification
Entity Extraction
Translation
Topic Modelling
...

Extract words from documents



Document $D = D_1, D_2, D_3, ..., D_m$

Word $W = W_1, W_2, W_3, ..., W_m$

Make vector by tf-idf method

Vector space model

$W=[W_1, W_2, W_3, ..., W_j, ..., W_n]$

$D =[D_1, D_2, D_3, ..., D_i, ..., D_m] =$

$$\begin{pmatrix} \alpha_{11} & \alpha_{12} & \cdots & \alpha_{1j} & \cdots & \alpha_{1n} \\ \alpha_{21} & \alpha_{22} & \cdots & \alpha_{2j} & \cdots & \alpha_{2n} \\ \vdots & \vdots & & \vdots & & \vdots \\ \alpha_{i1} & \alpha_{i2} & \cdots & \alpha_{ij} & \cdots & \alpha_{in} \\ \vdots & \vdots & & \vdots & & \vdots \\ \alpha_{m1} & \alpha_{m2} & \cdots & \alpha_{mj} & \cdots & \alpha_{mn} \end{pmatrix}$$

Weight of word W_i of document D_i

Weight vector of document "D_i"

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.

	T_1	T_2	....	T_t
D_1	w_{11}	w_{21}	…	w_{t1}
D_2	w_{12}	w_{22}	…	w_{t2}
:	:	:		:
:	:	:		:
D_n	w_{1n}	w_{2n}	…	w_{tn}

Term Weights: Term Frequency

The image cannot be displayed. Your computer may not have enough memory to open the image, or the image may have been corrupted. Restart your computer, and then open the file again. If the red x still appears, you may have to delete the image and then insert it again.

- 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*:

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

- 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)

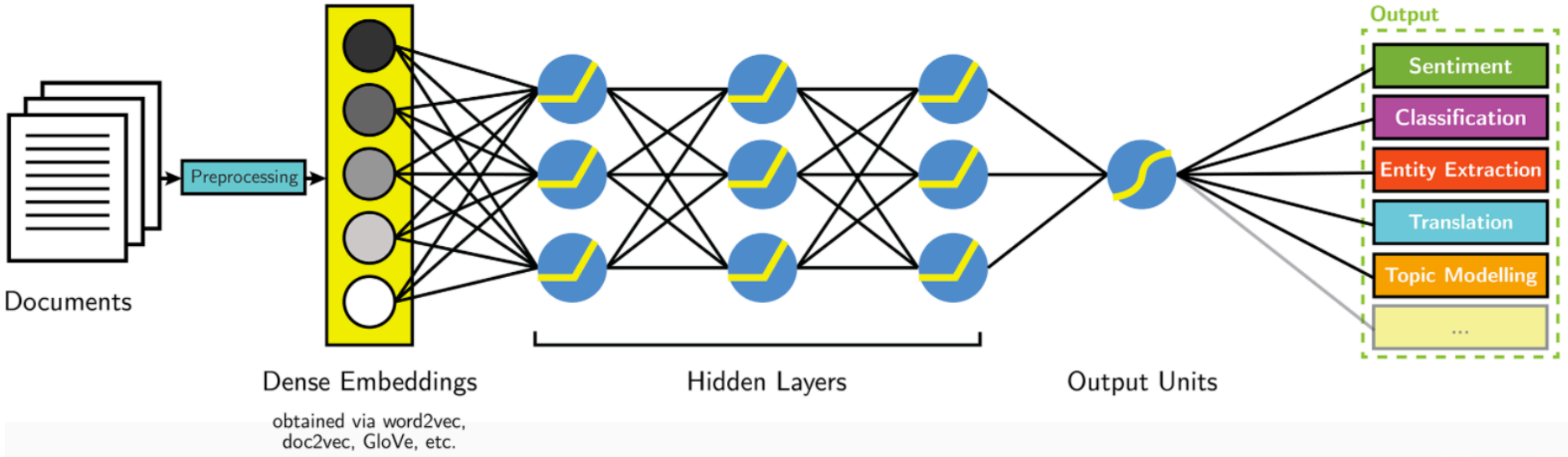
Then:

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

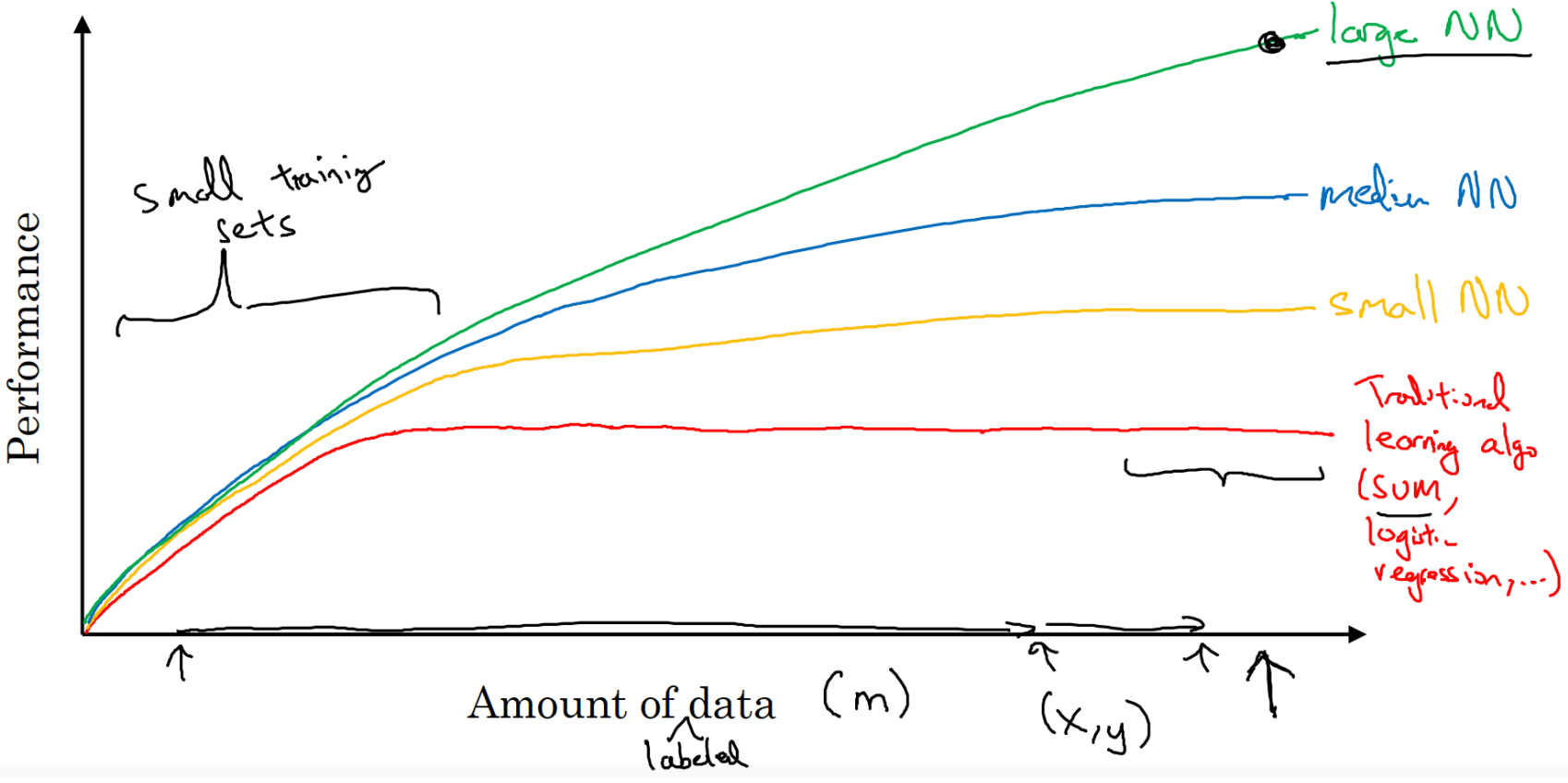
B: tf = 2/3; idf = $\log_2(10000/1300)$ = 2.9; tf-idf = 2.0

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

Neural-based Approaches



Why Neural?


large NN
medium NN
small NN
Traditional learning algo (SVM, logistic regression, ...)
Small training sets
Performance
Amount of data (m)
labeled
(x, y)

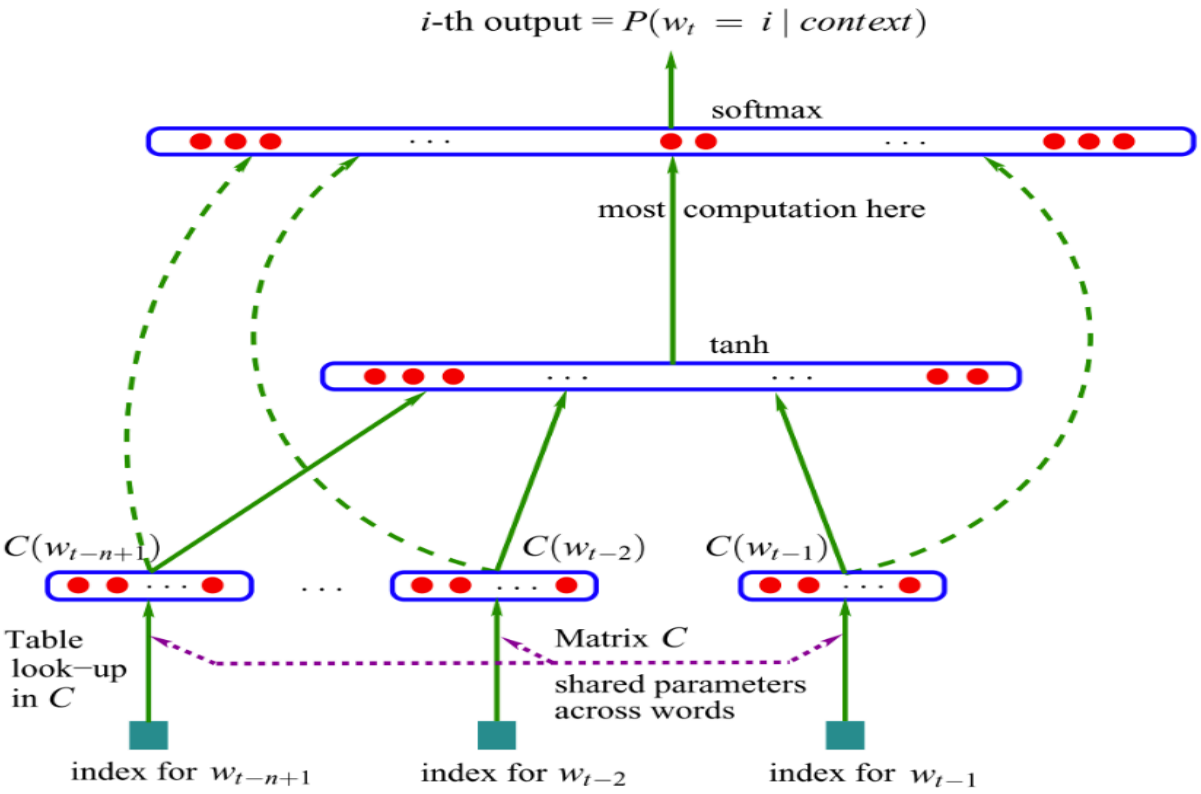
Neural-based Milestones for NLP

- 2001 - Neural language models
- 2008 - Multi-task learning
- 2013 - Word embeddings
- 2013 - Neural networks for NLP
- 2014 - Sequence-to-sequence models
- 2015 - Memory-based networks
- 2018 - Pretrained language models

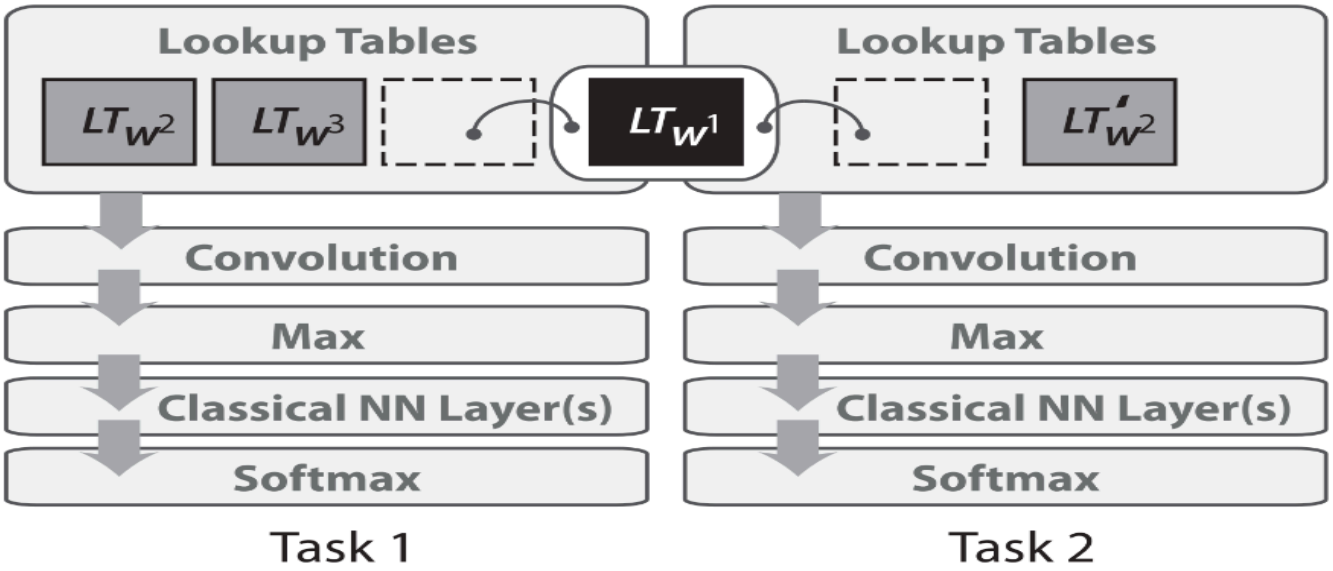
Character-based representations
Adversarial learning
Reinforcement learning

Conditional random fields
Bleu
Latent Dirichlet Allocation
Wikipedia exploitation

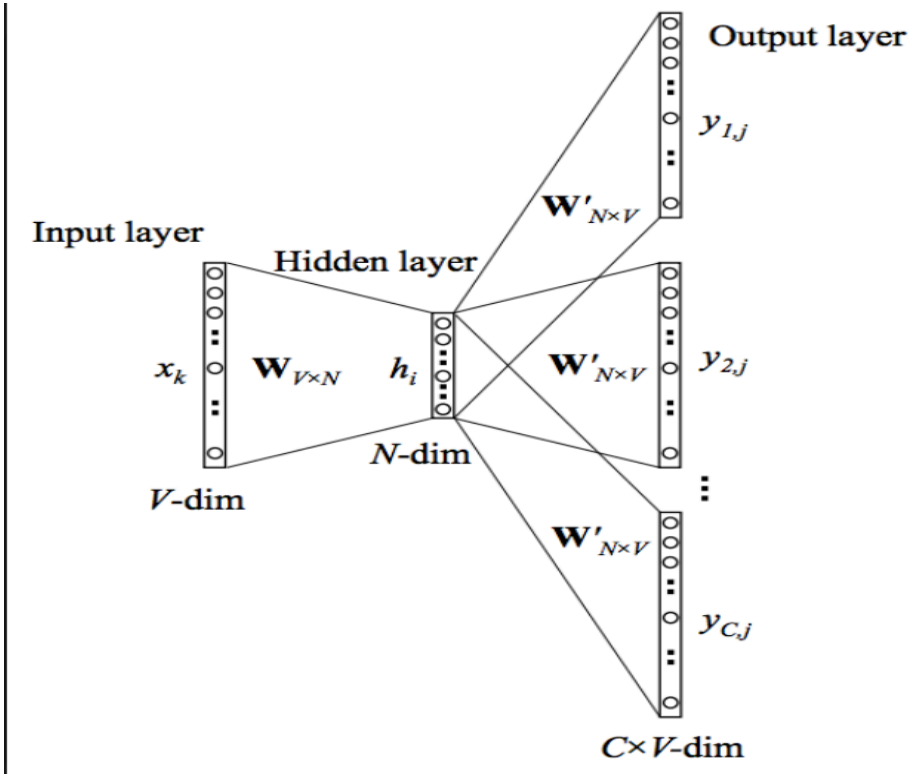
Neural Language Model

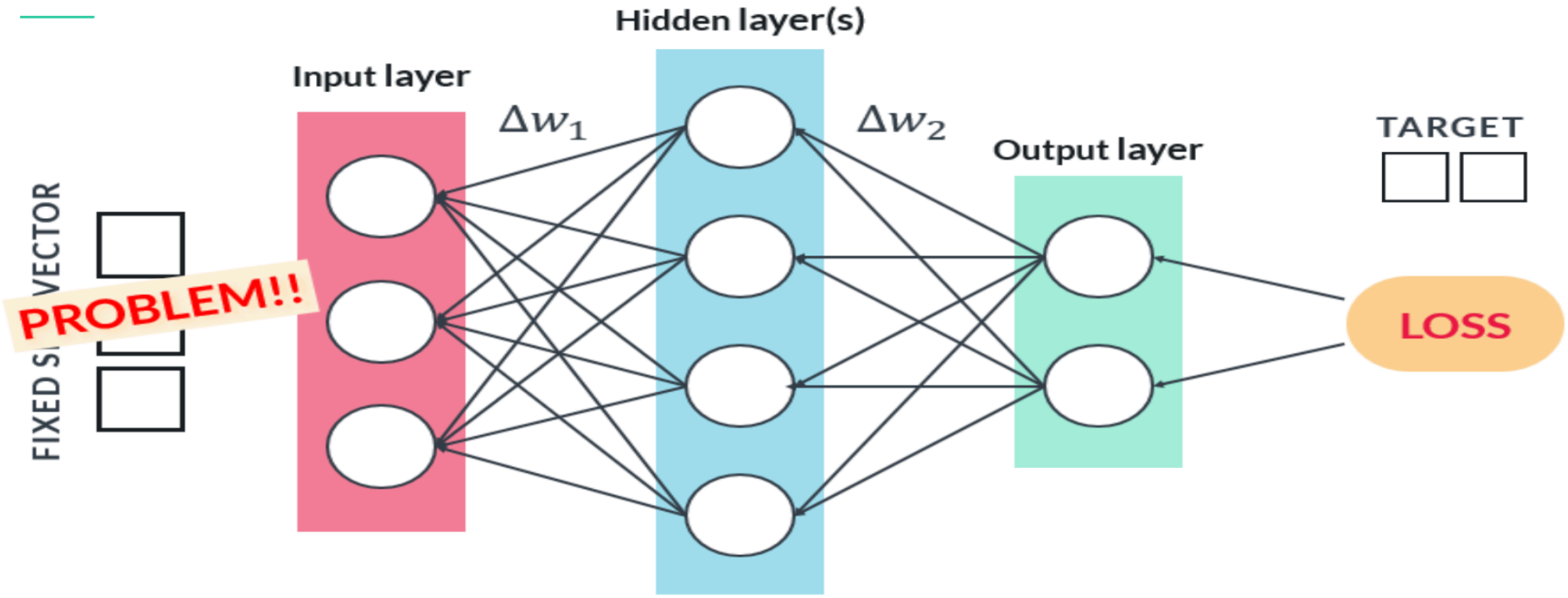


Multitask Learning

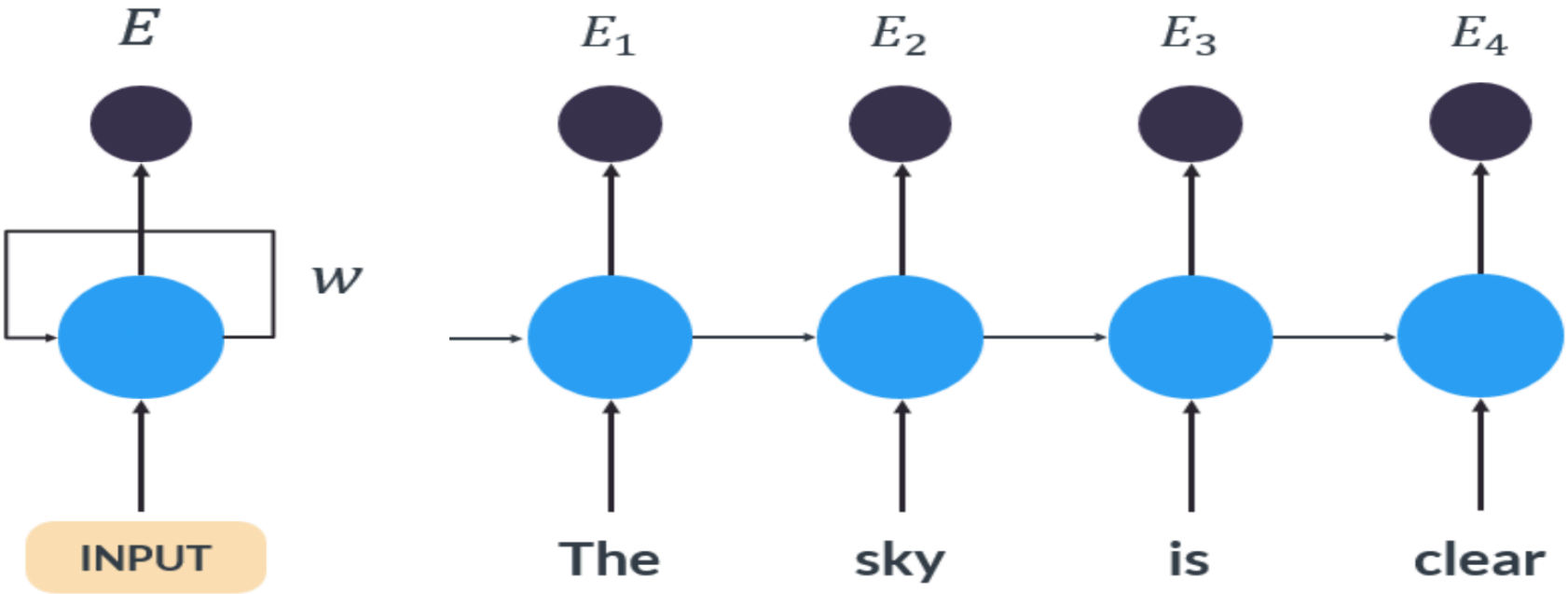


Word Embedding



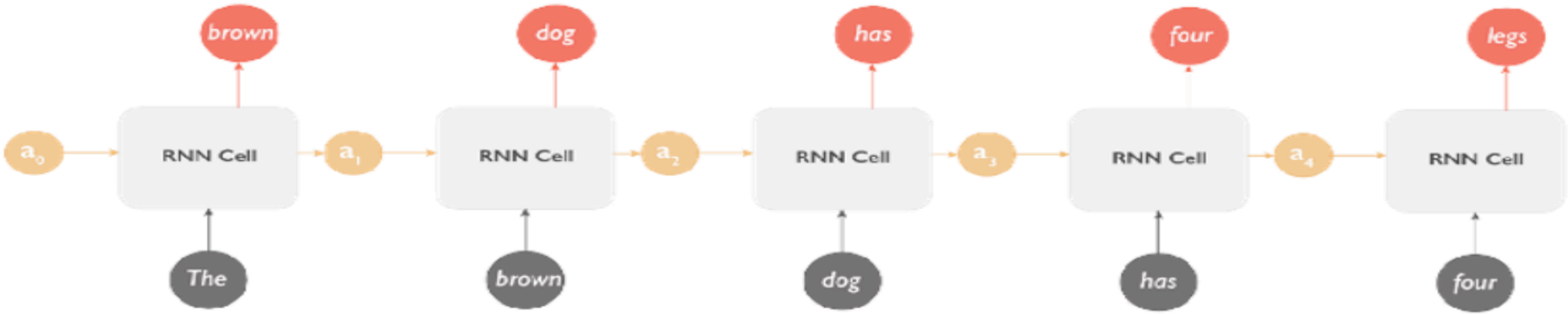

Hidden layer(s)
Input layer
Δw_1
Δw_2
Output layer
TARGET
FIXED S VECTOR
PROBLEM!!
LOSS

Recurrent Neural Networks



RNN Common Architectures

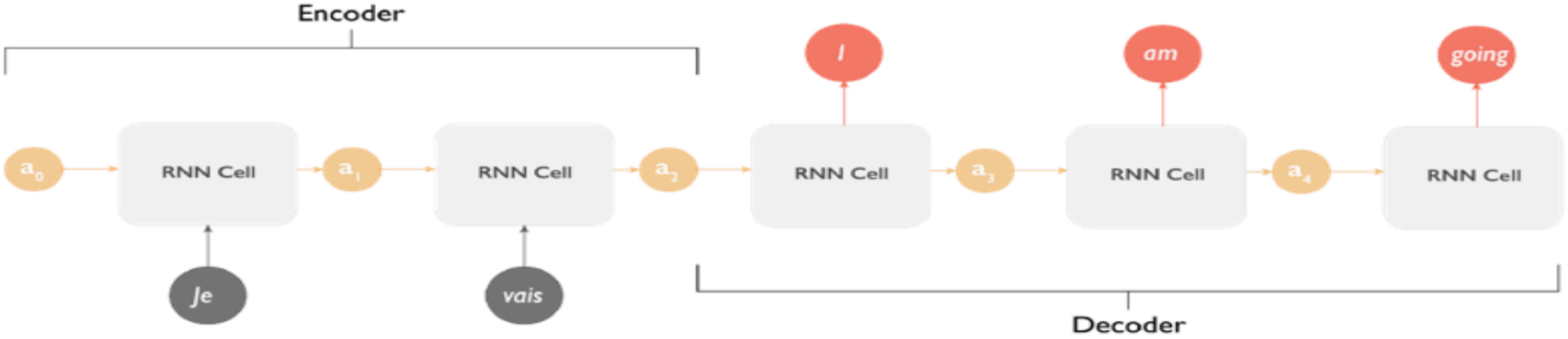
- **Many-to-many (same sequence lengths)**



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

RNN Common Architectures

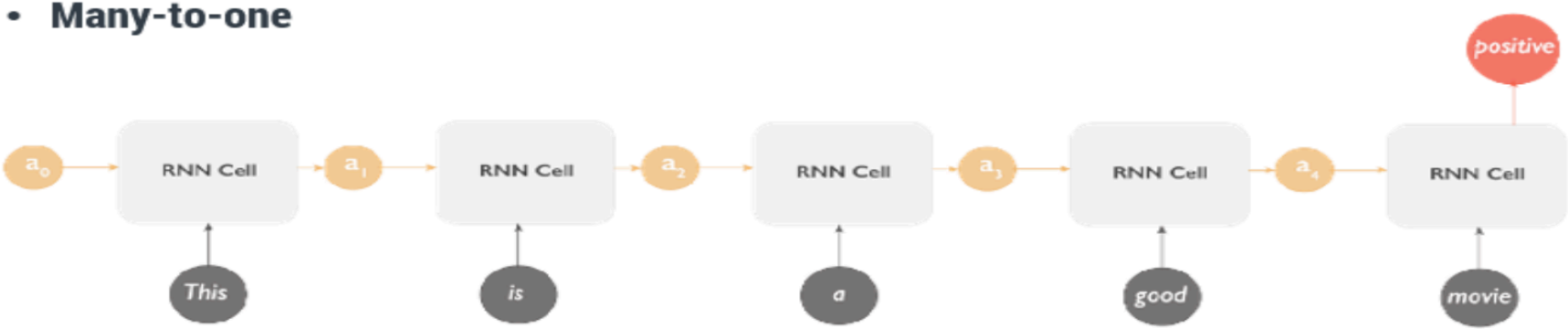
- **Many-to-many (same sequence lengths)**
- **Many-to-many (different sequence lengths)**



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

RNN Common Architectures

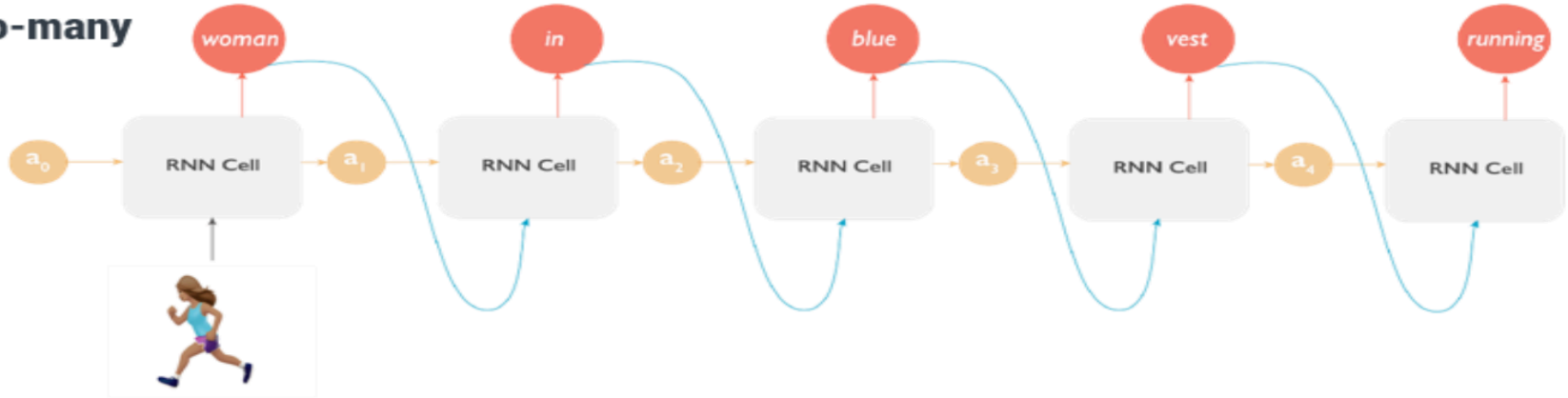
- **Many-to-many (same sequence lengths)**
- **Many-to-many (different sequence lengths)**
- **Many-to-one**



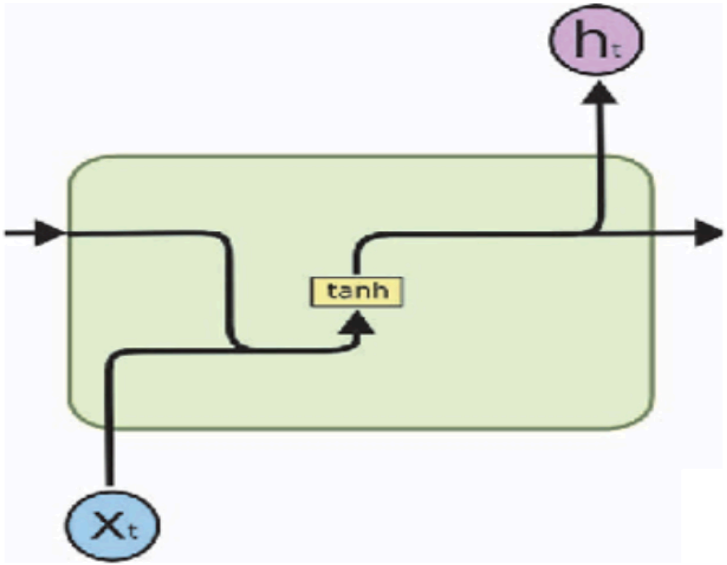
- **One-to-many**

RNN Common Architectures

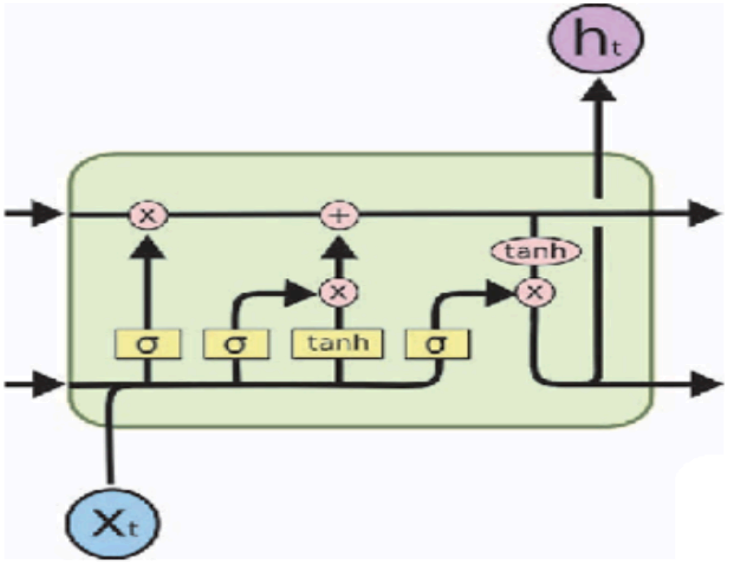
- Many-to-many (same sequence lengths)
- Many-to-many (different sequence lengths)
- Many-to-one
- One-to-many



Enhancement from RNN

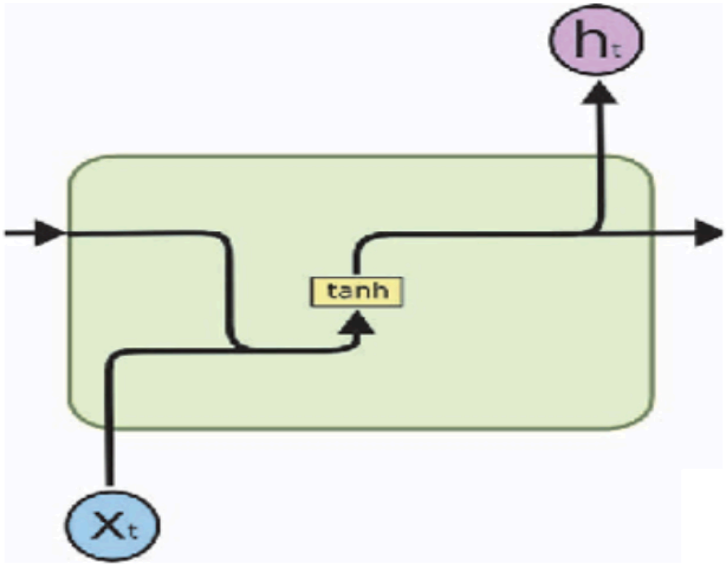


RNN

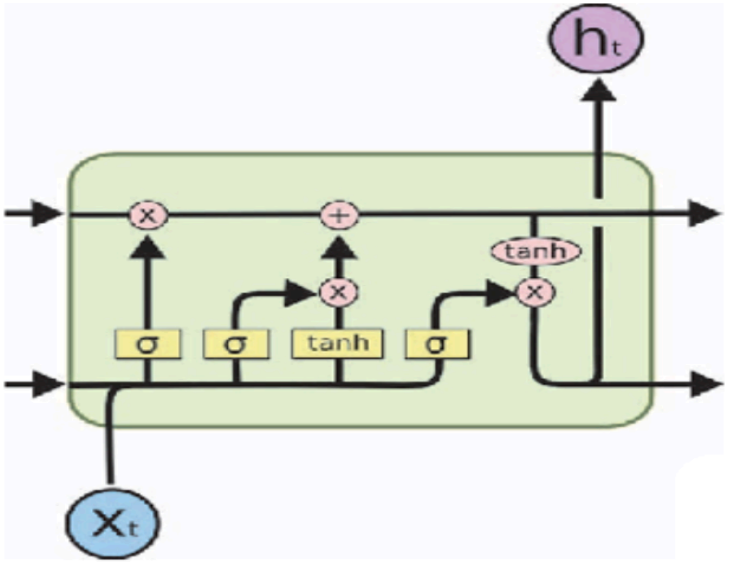


LSTM

Enhancement from RNN

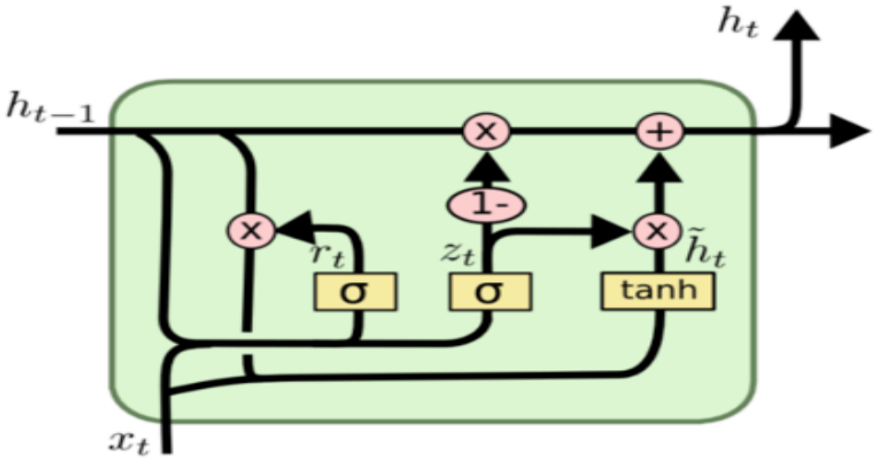


RNN



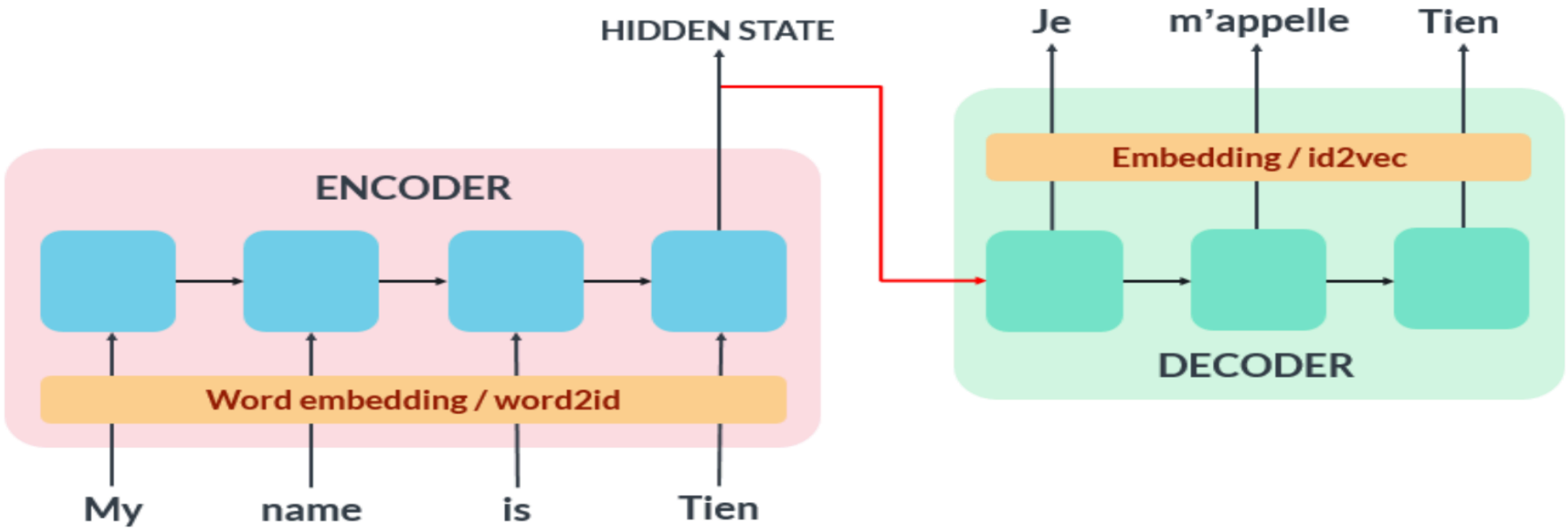
LSTM

Gated Recurrent Unit

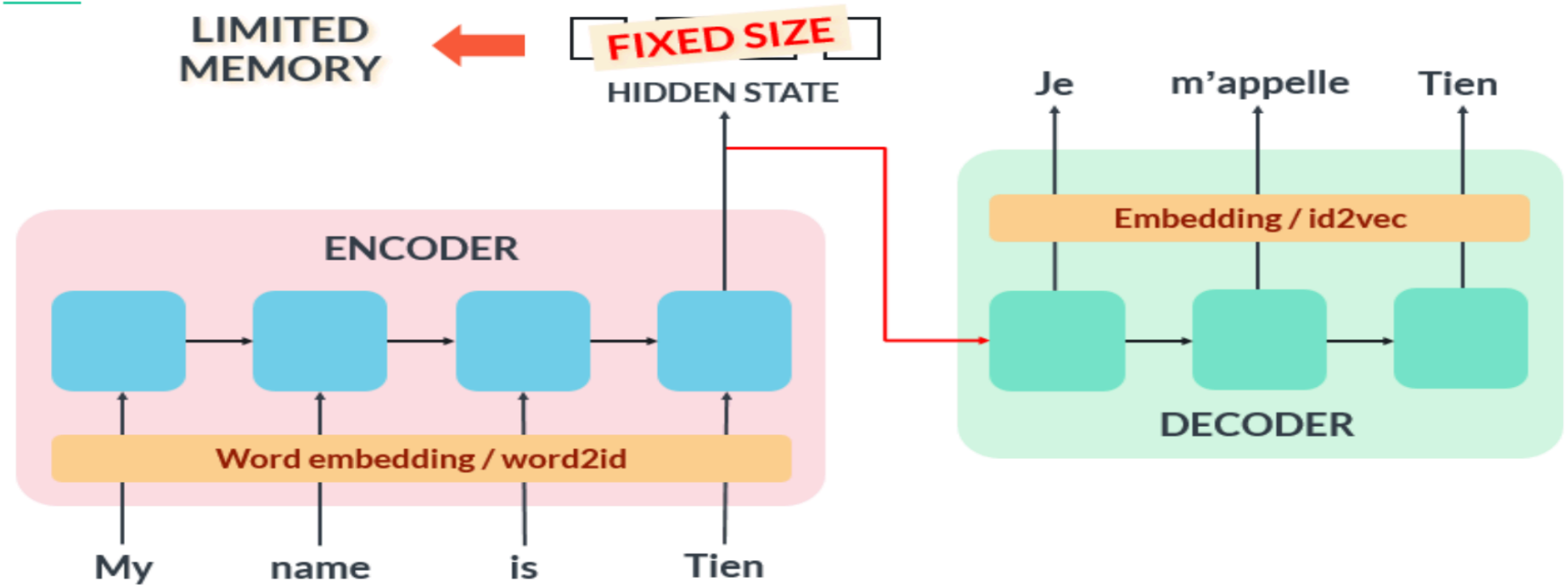


- Similar to LSTM but with **simpler** structure.
- Combine **forget** & **input gate** into ***"update gate".***
- Also combine **output gate** & **hidden state**.

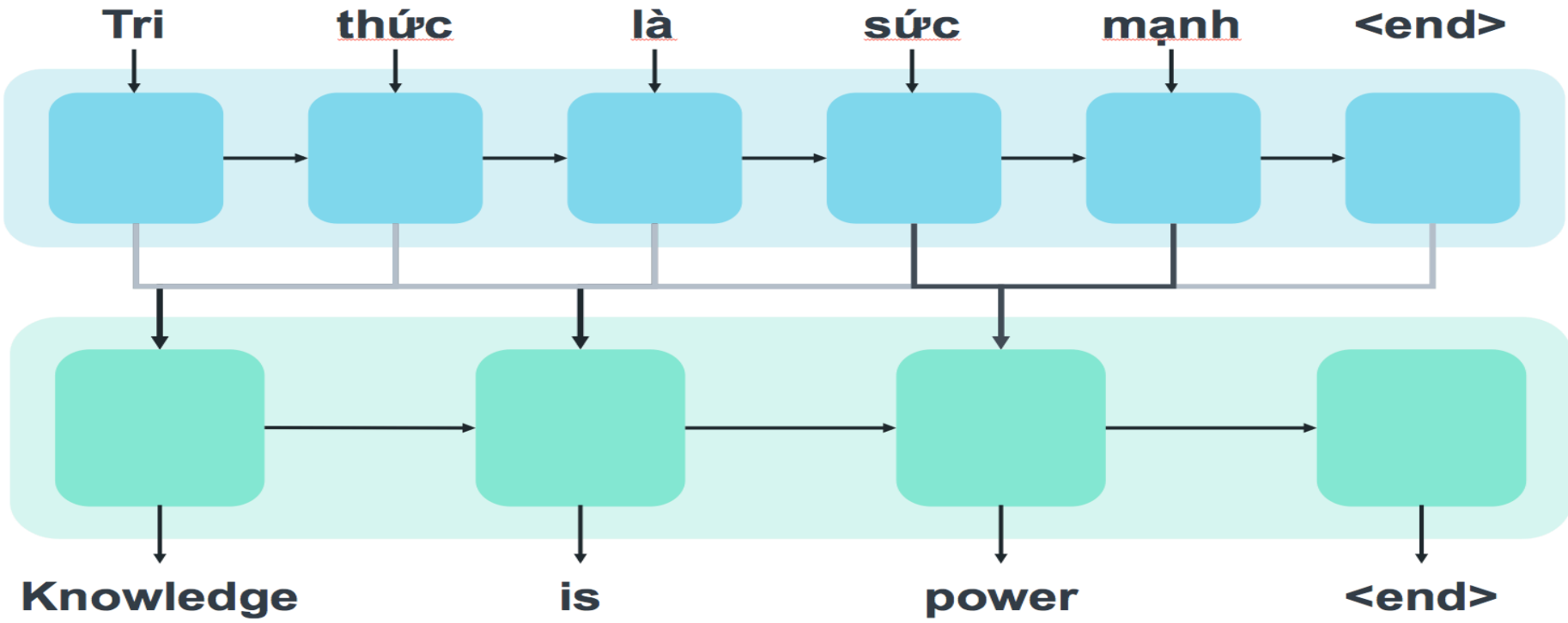
Seq2seq Architecture



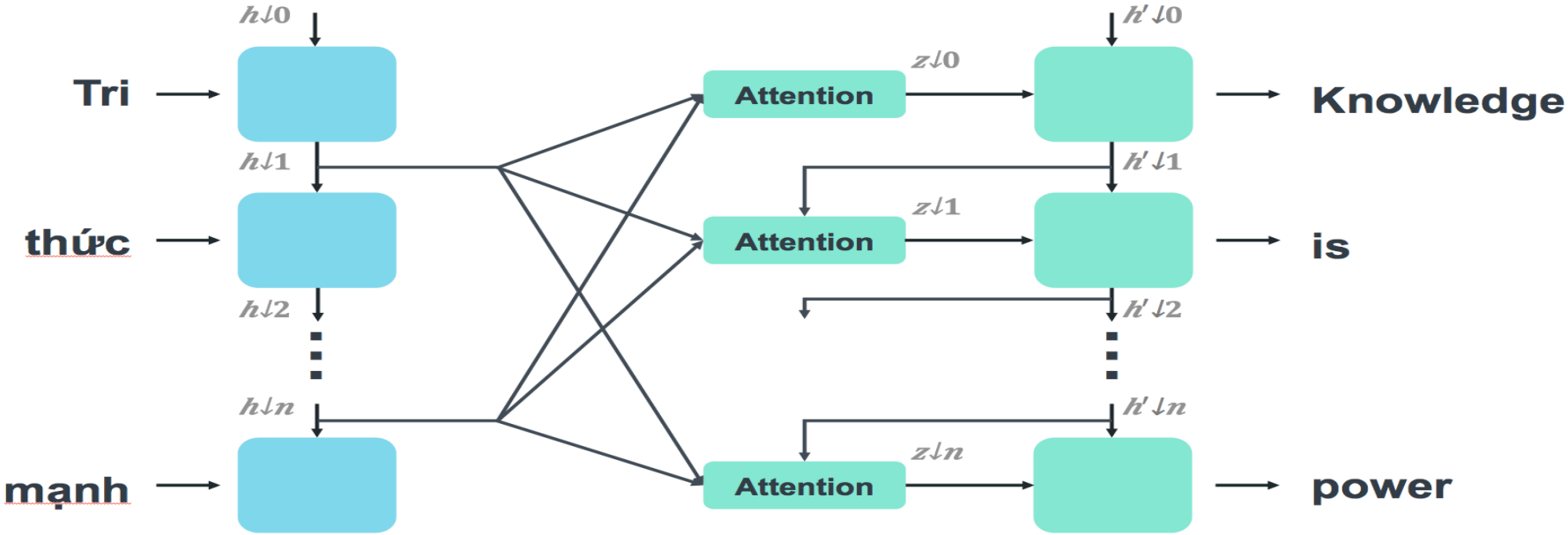
Seq2seq limitation



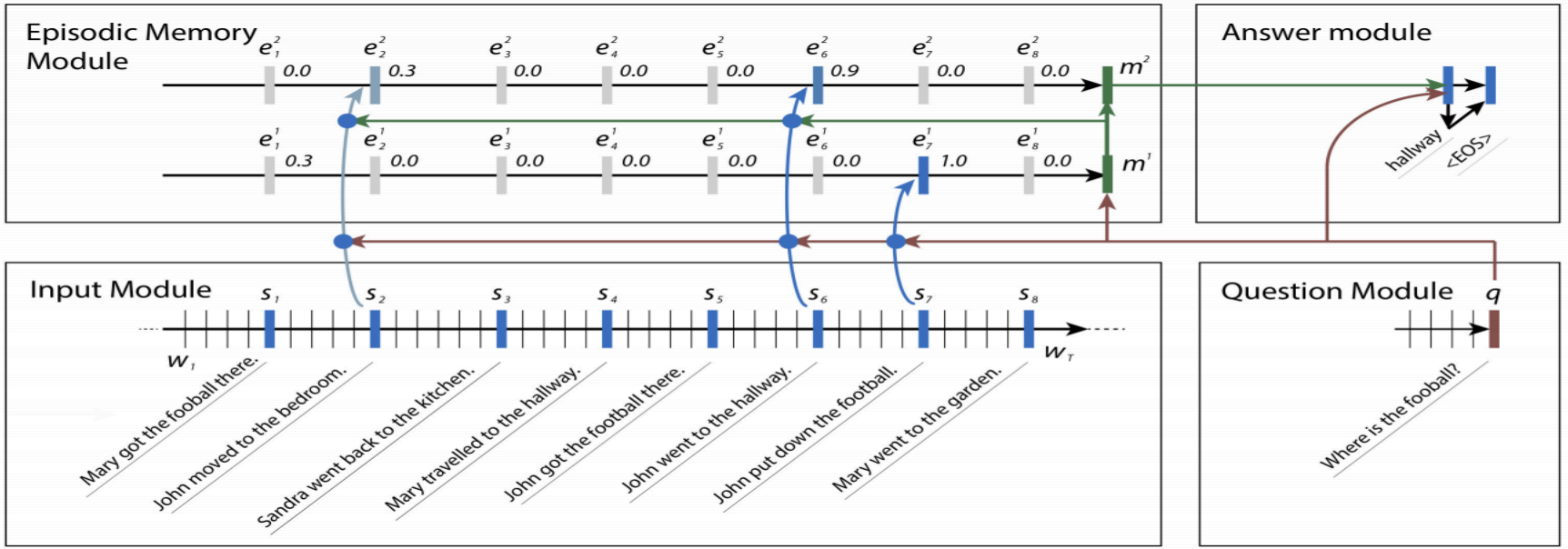
Attention Mechanism



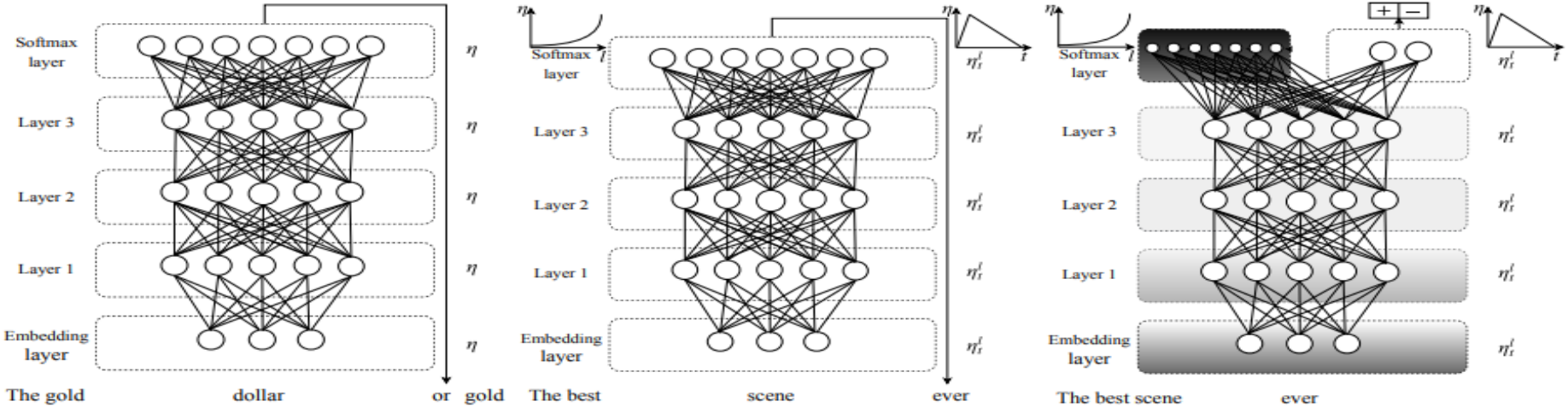
Attention Mechanism



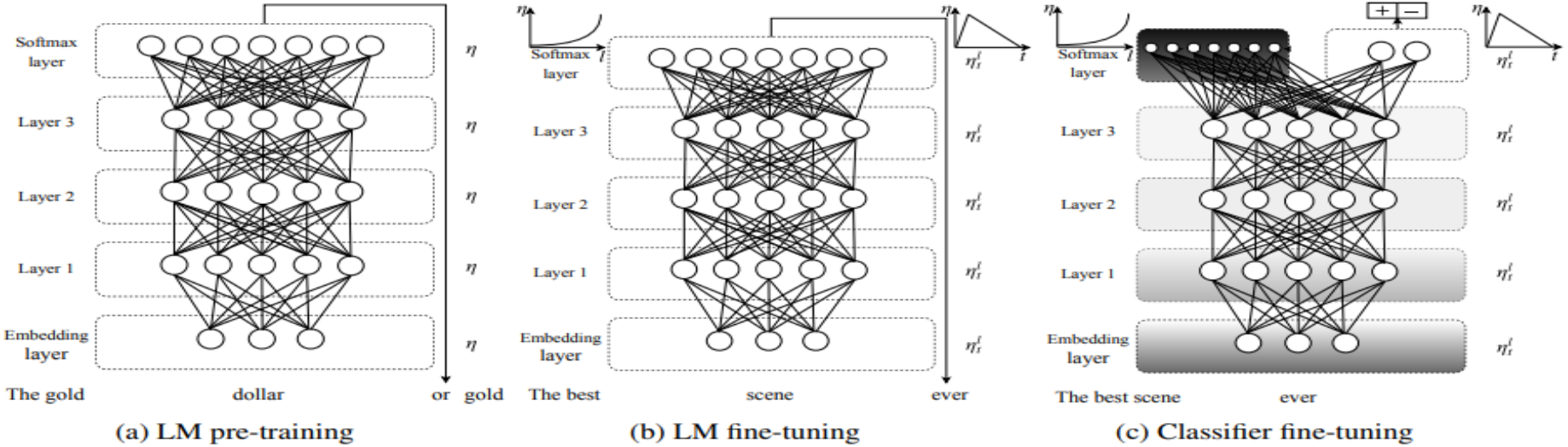
Dynamic Memory Model

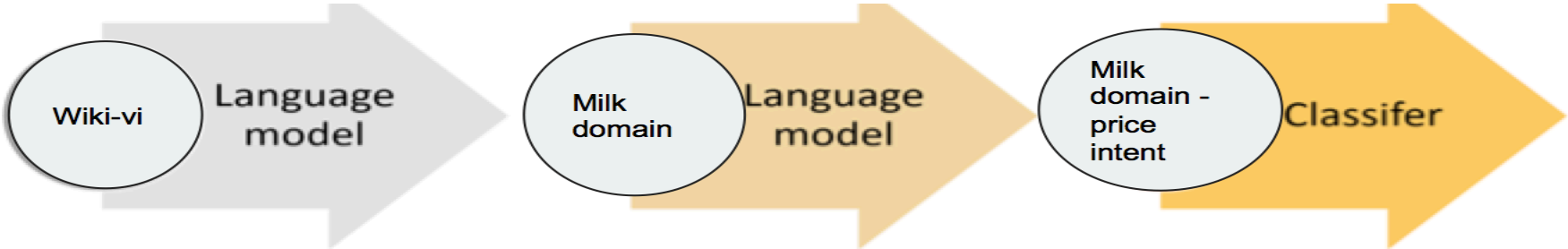


Pre-trained Language Model



Pre-trained Neural Language Model





Supervised Learning Problem

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

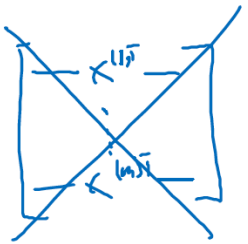
m training examples: $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})\}$

$m = m_{train}$ $\qquad$ m_{test} = #test examples.

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

n_x

m



$X \in \mathbb{R}^{n_x \times m}$ $\qquad$ $X.shape = (n_x, m)$

$$Y = \begin{bmatrix} y^{(1)} & y^{(2)} & \cdots & y^{(m)} \end{bmatrix}$$

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

$Y.shape = (1, m)$

Logistic Regression

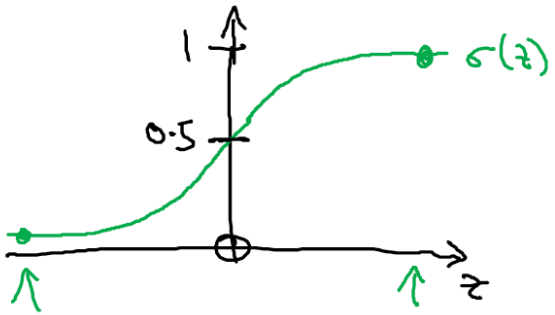
Given x, want $\hat{y} = P(y=1|x)$

$0 \le \hat{y} \le 1$

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

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

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



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

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

$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_{n_x} \end{bmatrix}$ $\}b \leftarrow$ $\}w \leftarrow$

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

If z 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

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

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

$x^{(i)}$ $y^{(i)}$ $z^{(i)}$ — i-th example.

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

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

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

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

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

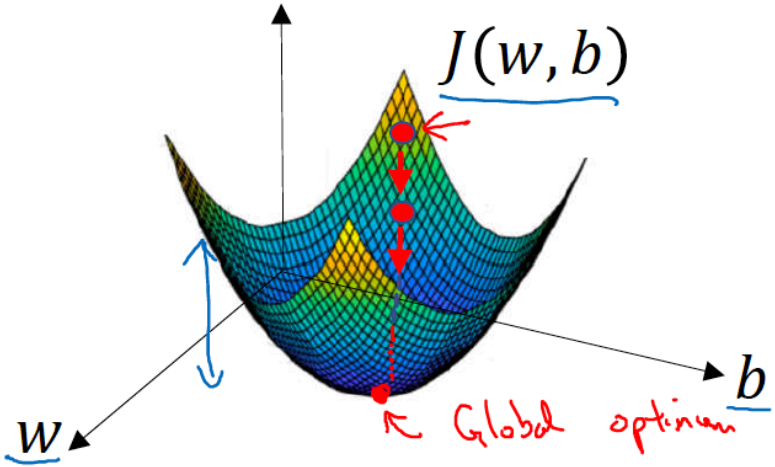
Cost function: $J(w,b) = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(\hat{y}^{(i)}, y^{(i)}) = -\frac{1}{m} \sum_{i=1}^{m} \left[y^{(i)} \log \hat{y}^{(i)} + (1-y^{(i)}) \log(1-\hat{y}^{(i)}) \right]$

Gradient Descent

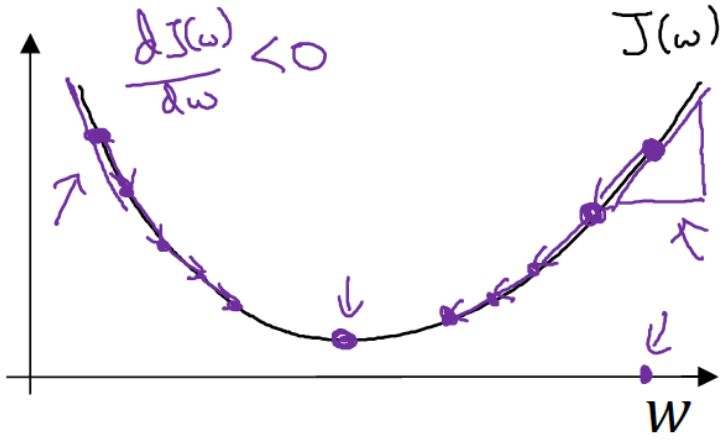
Recap: $\hat{y} = \sigma(w^T x + b)$, $\sigma(z) = \frac{1}{1+e^{-z}}$ ←

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

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



Gradient Descent



Repeat {

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

learning rate

"dw"

$w := w - \alpha dw$

}

$\frac{dJ(w)}{dw} = ?$

$J(w,b)$

$w := w - \alpha \frac{\partial J(w,b)}{\partial w}$

$b := b - \alpha \frac{\partial J(w,b)}{\partial b}$

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

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

∂ "partial derivative"

J

Computation Graph

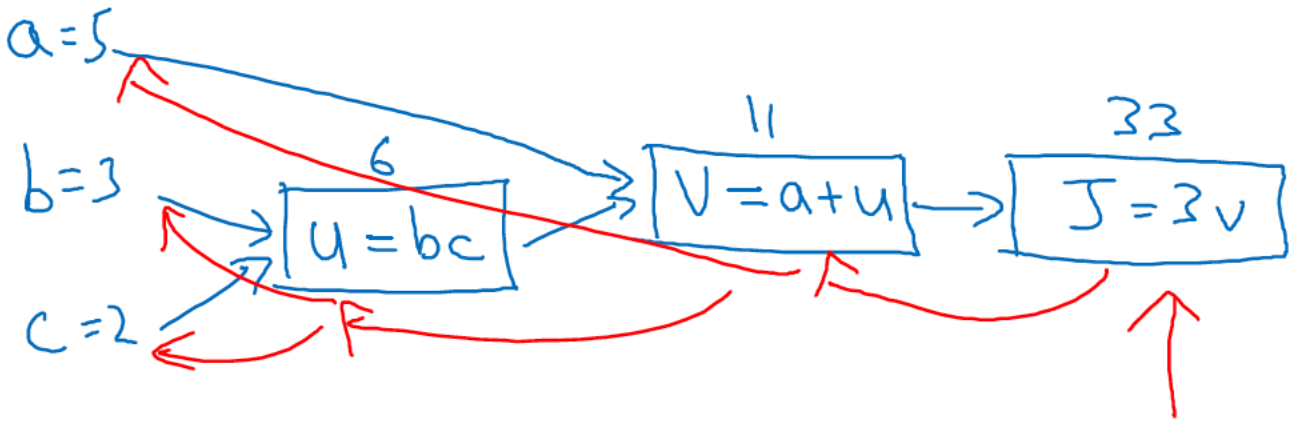
$J(a,b,c) = 3(a+bc) = 3(5+3\times 2) = 33$

u → v → J

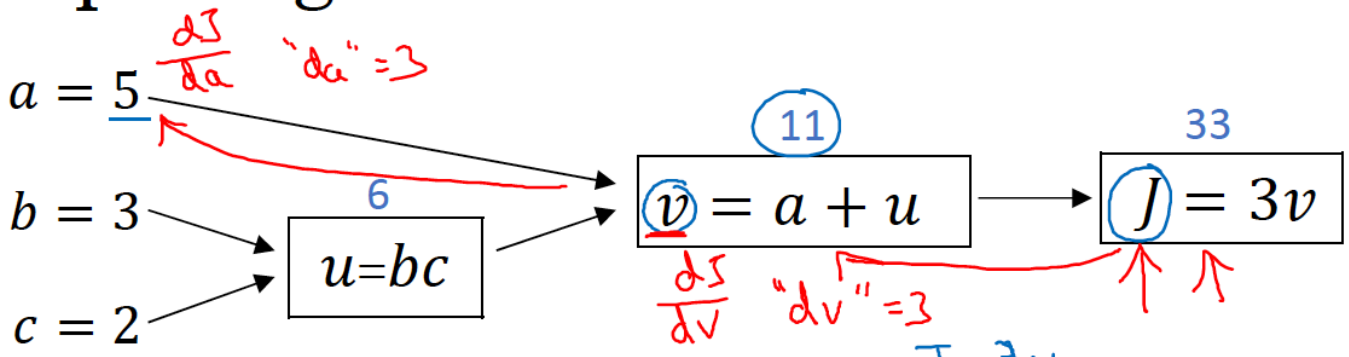
$u = bc$

$v = a+u$

$J = 3v$



Computing derivatives



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

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

3×1

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

$a \to v \to J$

$J = 3v$

$v = 11 \to 11.001$

$J = 33 \to 33.003$

$a = 5 \to 5.001$

$\to v = 11 \to 11.001$

$J = 33 \to 33.003$

$\frac{d\,\text{FinalOutputVar}}{d\,\text{var}}$ → "dvar"

dJ dvar

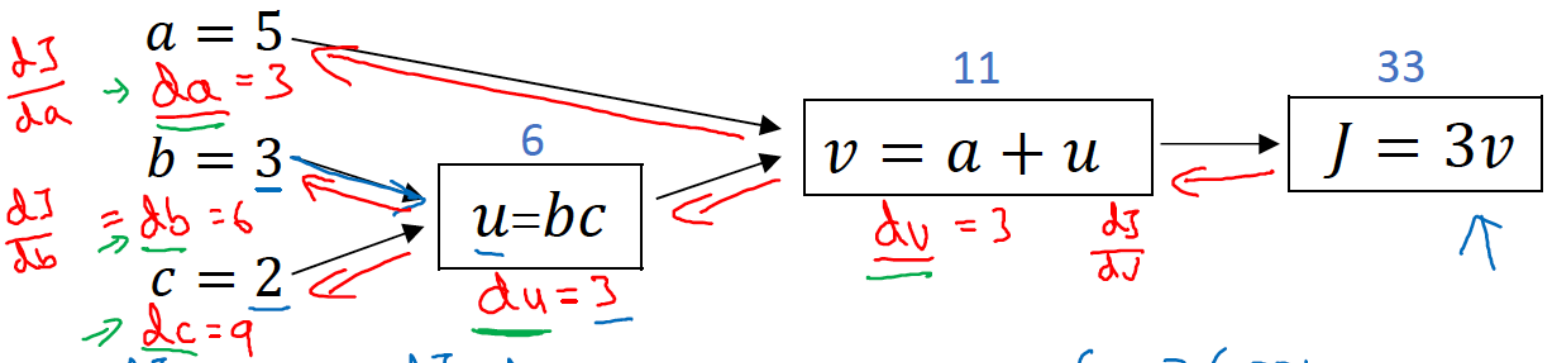
$f(a) = 3a$

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

$J = 3v$

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

Computing derivatives



$\frac{dJ}{da} = 3 = \frac{dJ}{dv} \cdot \frac{dv}{du}$ (3 · 1)

$\frac{dJ}{db} = \frac{dJ}{du} \cdot \frac{du}{db} = 6$ ($\frac{dJ}{du} = 3$, $\frac{du}{db} = 2$)

$\frac{dJ}{da} = \frac{dJ}{du} \cdot \frac{du}{da} = 9$ (3 × 3)

$u = 6 \rightarrow 6.001$
$v = 11 \rightarrow 11.001$
$J = 33 \rightarrow 33.003$

$b = 3 \rightarrow 3.001$
$u = b \cdot c = 6 \rightarrow 6.002$
$J = 33.006$

$c = 2$
.006

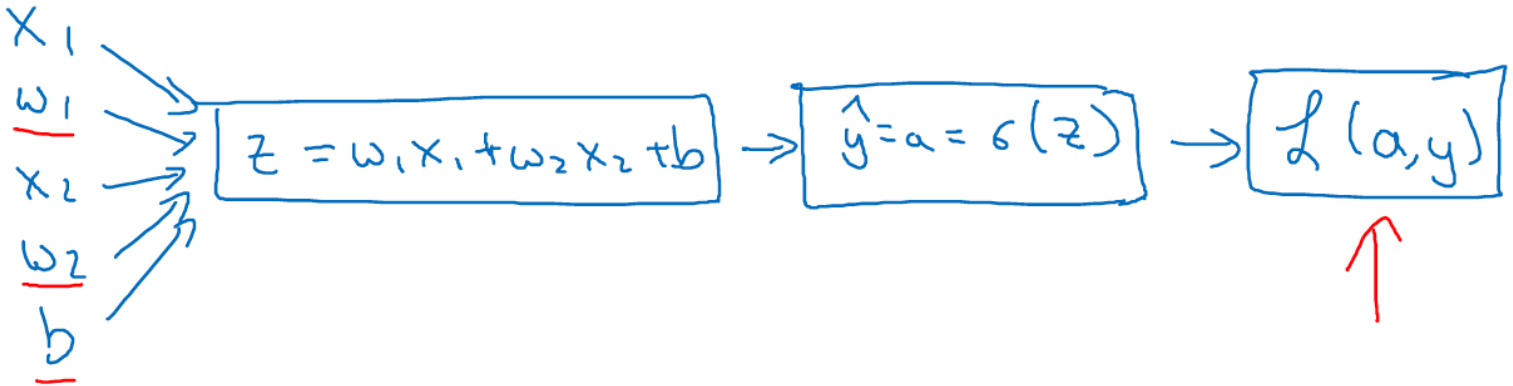
$V = 11.002$
$J = 3V$

Logistic regression recap

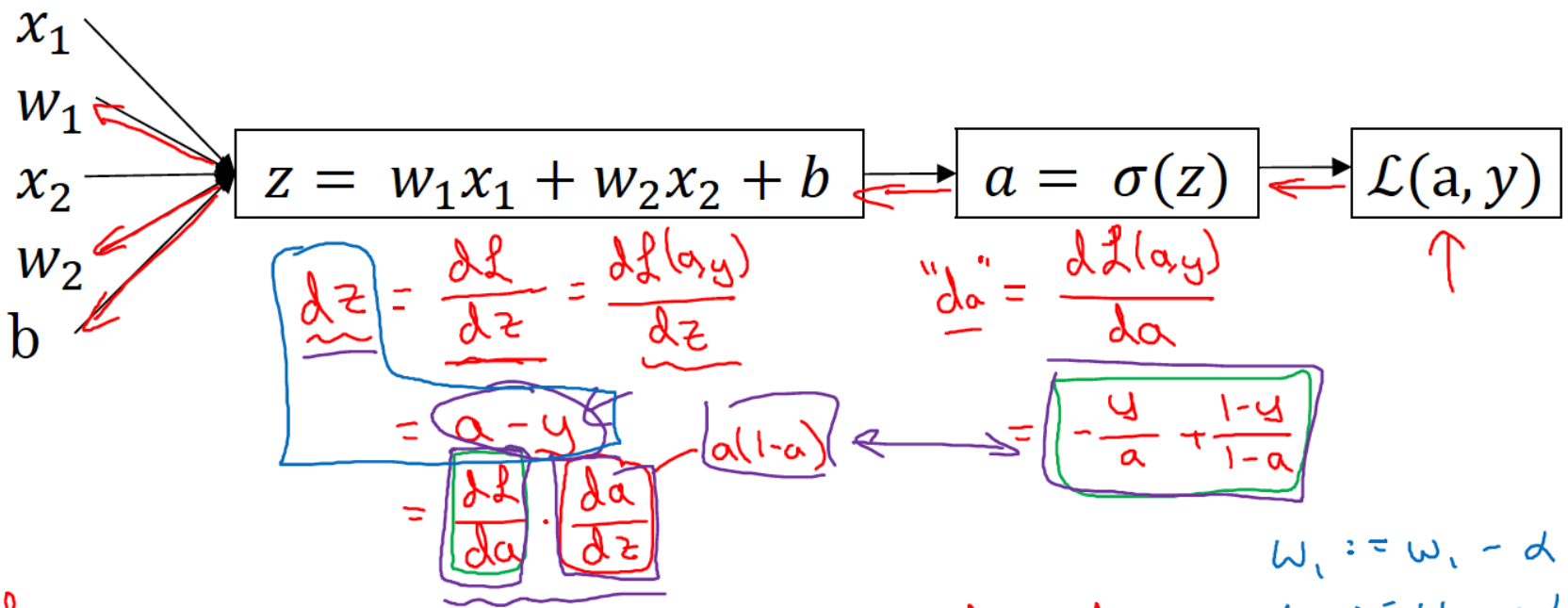
$$z = w^T x + b$$

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

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



Logistic regression derivatives



Logistic regression on m examples

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

$$\rightarrow a^{(i)} = \hat{y}^{(i)} = \sigma(z^{(i)}) = \sigma(w^T x^{(i)} + b)$$

$(x^{(i)}, y^{(i)})$

$dw_1^{(i)}$, $dw_2^{(i)}$, $db^{(i)}$

$$\frac{\partial}{\partial w_1} J(w,b) = \frac{1}{m}\sum_{i=1}^{m} \underbrace{\frac{\partial}{\partial w_1} \mathcal{L}(a^{(i)}, y^{(i)})}_{dw_1^{(i)} \text{ - } (x^{(i)}, y^{(i)})}$$

Logistic regression on m examples

$J = 0$; $dw_1 = 0$; $dw_2 = 0$; $db = 0$

For $i = 1$ to m

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

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

$J \mathrel{+}= -\left[y^{(i)} \log a^{(i)} + (1 - y^{(i)}) \log(1 - a^{(i)})\right]$

$dz^{(i)} = a^{(i)} - y^{(i)}$

$dw_1 \mathrel{+}= x_1^{(i)} dz^{(i)}$

$dw_2 \mathrel{+}= x_2^{(i)} dz^{(i)}$

dw_3 ... dw_n

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

$n = 2$

$J /= m$

$dw_1 /= m$; $dw_2 /= m$; $db /= m$.

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

$w_1 := w_1 - \alpha\, dw_1$

$w_2 := w_2 - \alpha\, dw_2$

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

Vectorization

Vectorizing Logistic Regression

$$z^{(1)} = w^T x^{(1)} + b$$
$$a^{(1)} = \sigma(z^{(1)})$$

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

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

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

(n_x, m)

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

$$w^T \begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ | & | & & | \end{bmatrix}$$

$$Z = \begin{bmatrix} z^{(1)} & z^{(2)} & \cdots & z^{(m)} \end{bmatrix} = w^T X + \underbrace{[b \; b \cdots b]}_{1 \times m} = \begin{bmatrix} w^T x^{(1)} + b & w^T x^{(2)} + b & \cdots & w^T x^{(m)} + b \end{bmatrix}$$

$1 \times m$

$$Z = np.dot(w.T, X) + b$$

$(1,1)$ $\mathbb{R}$

"Broadcasting"

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

Vectorizing Logistic Regression

$dz^{(1)} = a^{(1)} - y^{(1)}$ $\quad dz^{(2)} = a^{(2)} - y^{(2)}$ $\quad \cdots$

$dZ = [dz^{(1)}\ dz^{(2)} \cdots dz^{(m)}]$ $\quad (1 \times m)$ $\leftarrow$

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

$\rightarrow dZ = A - Y = [a^{(1)} - y^{(1)} \quad a^{(2)} - y^{(2)} \quad \cdots]$

$\rightarrow dw = 0$

$dw \mathrel{+}= x^{(1)} dz^{(1)}$

$dw \mathrel{+}= x^{(2)} dz^{(2)}$

$\vdots$

$dw /= m$

$db = 0$

$db \mathrel{+}= dz^{(1)}$

$db \mathrel{+}= dz^{(2)}$

$\vdots$

$db \mathrel{+}= dz^{(m)}$

$db /= m.$

$db = \frac{1}{m} \sum_{i=1}^{m} dz^{(i)}$

$= \frac{1}{m}$ np.sum(dZ)

$dw = \frac{1}{m} X dZ^T$

$= \frac{1}{m} \begin{bmatrix} x^{(1)} & \cdots & x^{(m)} \\ 1 & & 1 \end{bmatrix} \begin{bmatrix} dz^{(1)} \\ \vdots \\ dz^{(m)} \end{bmatrix}$

$= \frac{1}{m} [x^{(1)} dz^{(1)} + \cdots + x^{(m)} dz^{(m)}]$

$n \times 1$

Implementing Logistic Regression

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

for i = 1 to m:

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

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

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

$dz^{(i)} = a^{(i)} - y^{(i)}$

$dw_1\ += x_1^{(i)} dz^{(i)}$

$dw_2\ += x_2^{(i)} dz^{(i)}$

$dw\ += x^{(i)} * dz^{(i)}$

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

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

$db = db/m$

for iter in range(1000):

$Z = w^T X + b$

$= np.dot(w.T, X) + b$

$A = \sigma(Z)$

$dZ = A - Y$

$dw = \frac{1}{m} X dZ^T$

$db = \frac{1}{m} np.sum(dZ)$

$w := w - \alpha dw$

$b := b - \alpha db$