國立高雄科技大學
National Kaohsiung University of Science and Technology

Introduction to Natural Language Processing & Neural Network

報告人: 蘇佳益
指導老師: 陳聰毅
國立高雄科技大學建功校區電子工程系

課程資料

- 投影片 連結
  - https://github.com/chiayisu/Natural-Language-Processing
- 程式連結
  - https://github.com/oreilly-japan/deep-learning-from-scratch-2 (Book)
  - https://github.com/chiayisu/NLP_and_ML_Algorithm (My Implementation)
- 參考書籍
  - 斎藤康毅, Deep Learning 2: 用Python進行自然語言處理的基礎理論實作
- 參考課程
  - http://web.stanford.edu/class/cs224n/ (主要補充書中講比較少的內容)

Agenda

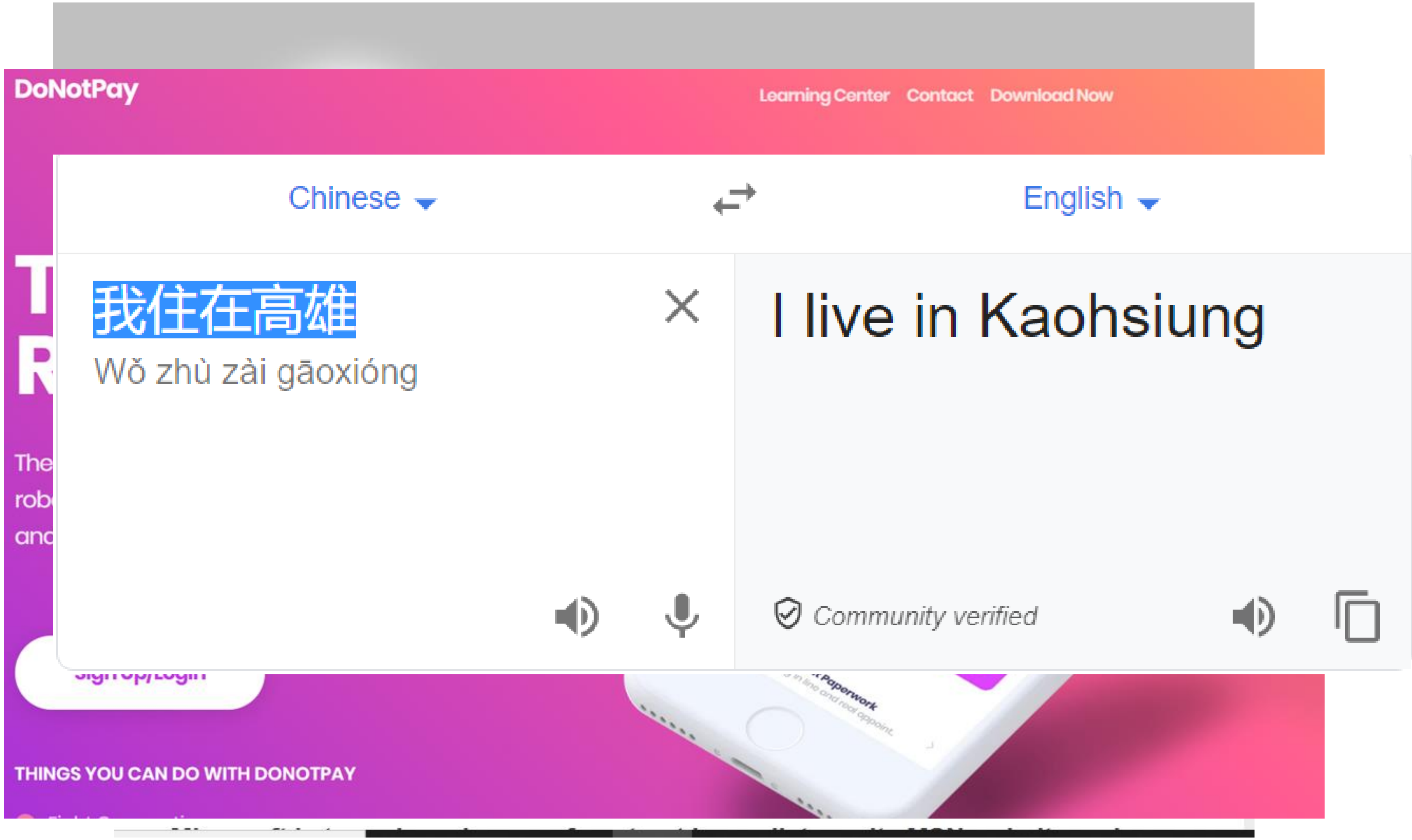
- 自然語言處理
- The Design and Implementation of XiaoIce, an Empathetic Social Chatbot
- Measuring Depression Symptom Severity from Spoken Language and 3D Facial Expressions
- Google Assistant
- 2017 Google I/O
- GPT-3 模型應用
- Visual 20 Question Games
- Neural Network
- 微分複習
- Useful Identities for Neural Network
- References

自然語言處理

甚麼是自然語言處理?

自然語言處理的應用

NLP is Everywhere



自然語言處理的演進

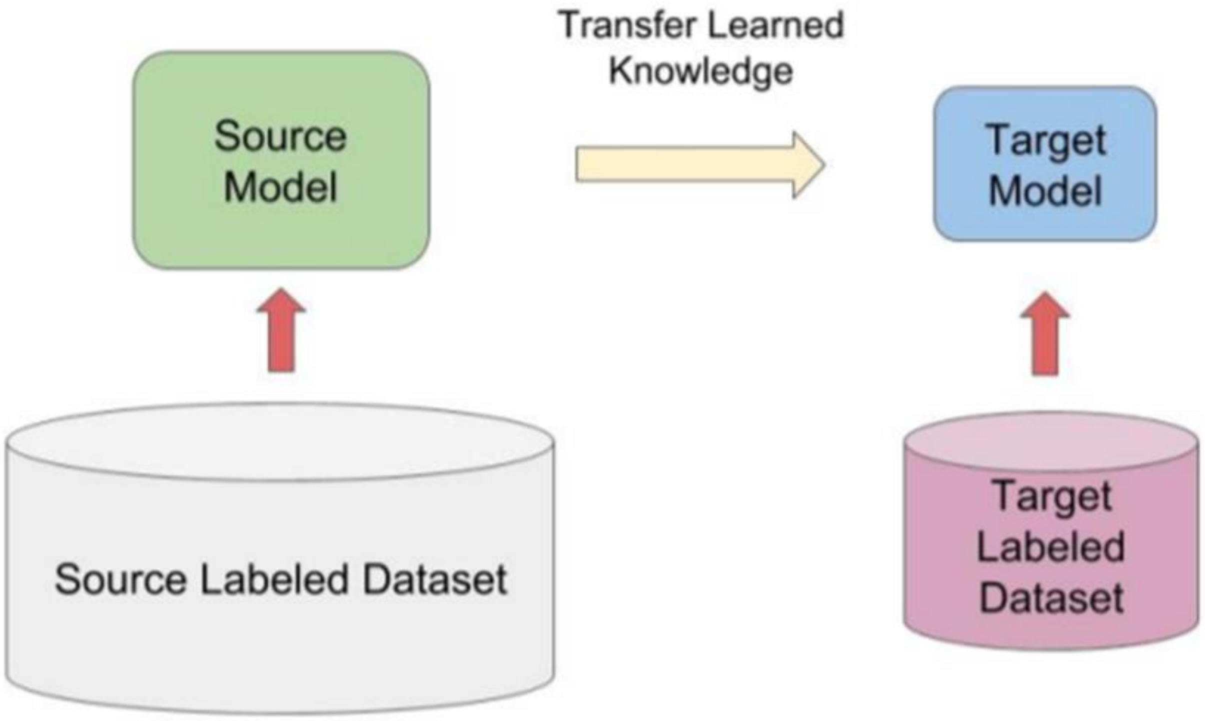
Count-Based Model → Word Representation → Contextual Representation

自然語言處理的演進

Co-Matrix	Word2Vec	GPT	BERT	GPT-2	GPT-3
	Mikolov et al., 2013.	Radford et al., 2018.	Devlin et al., 2018	Radford et al., 2019	Brown et al., 2020
	Google	OpenAI	Google AI	OpenAI	OpenAI

Contextual Representation (Transformer)

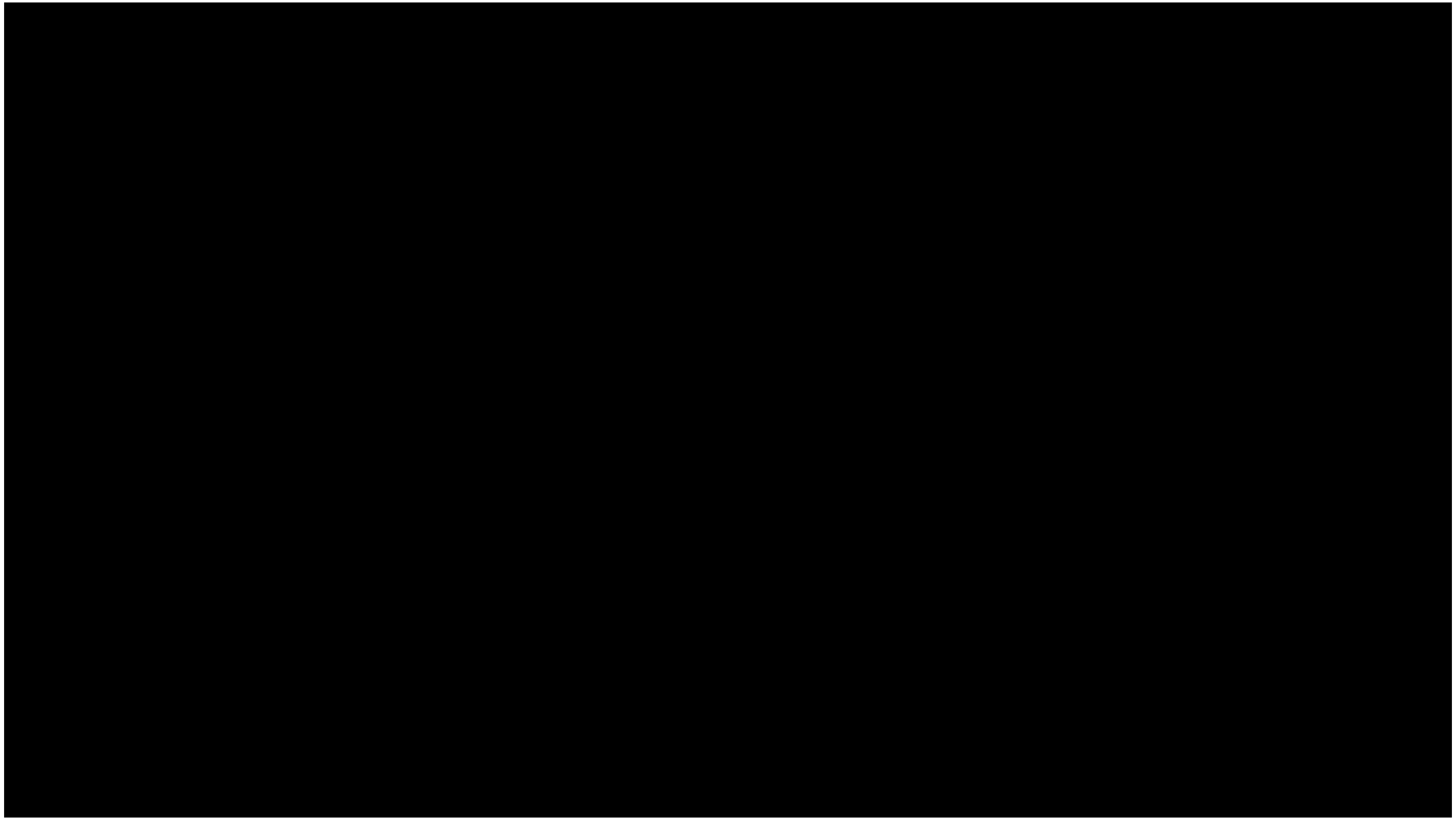
The State-of-the-Art Learning Method - 遷移學習



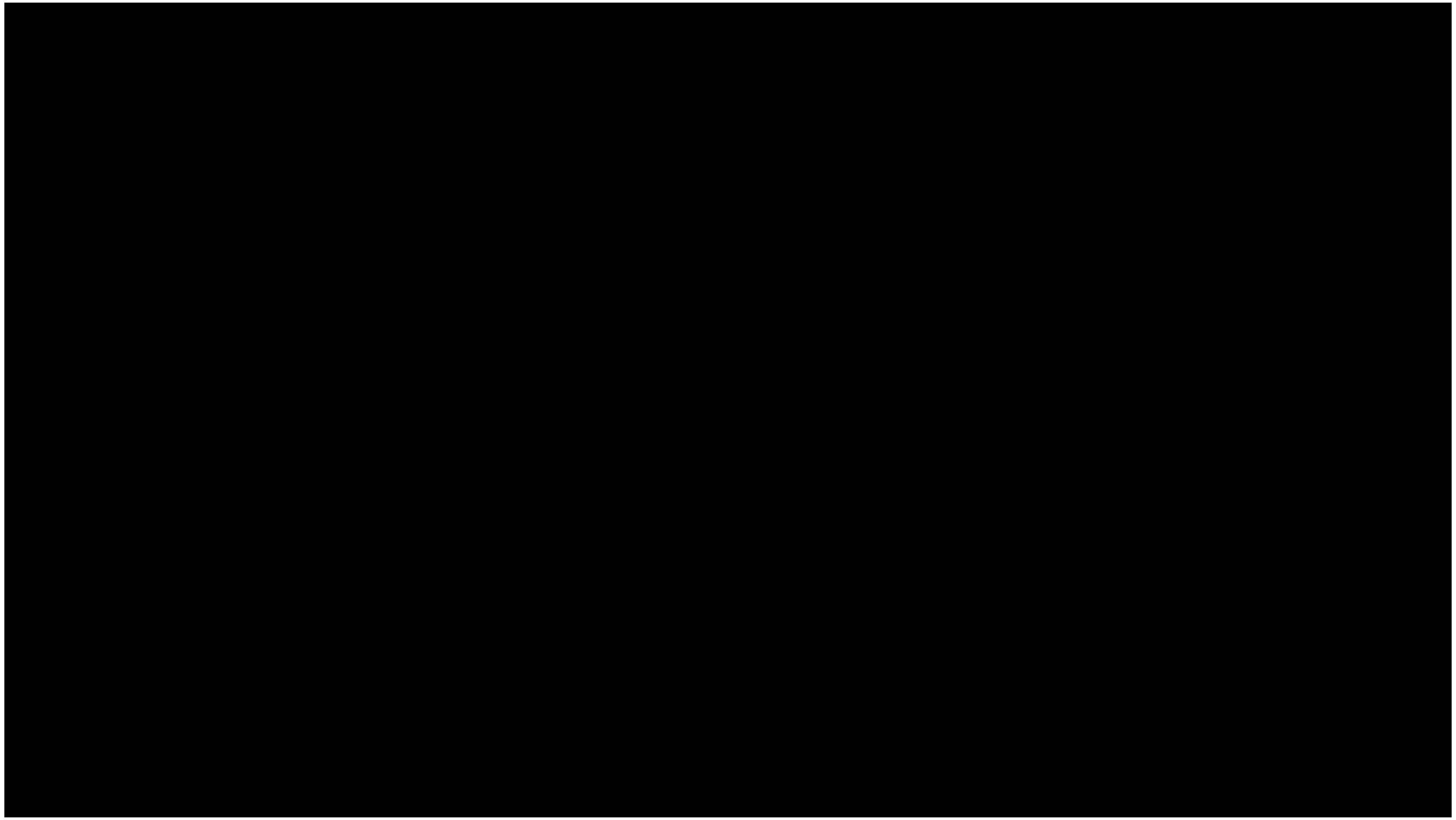
The Design and Implementation of XiaoIce, an Empathetic Social Chatbot

Li Zhou et al.

DEMO – 唱歌



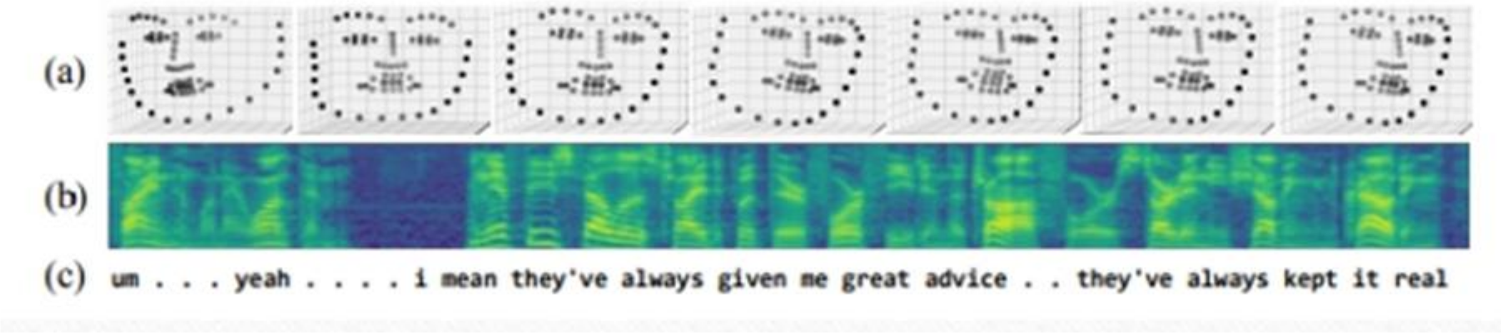
DEMO – 對話



Measuring Depression Symptom Severity from Spoken Language and 3D Facial Expressions

Haque et al.

Method



- Models
  - Word-Embedding
  - C-CNN

Google Assistant

2017 Google I/O

國立高雄科技大學
National Kaohsiung University of Science and Technology

GPT-3 模型應用

- HTML Layout 產生
  - https://twitter.com/i/status/1282676454690451457
- 網頁產生
  - https://twitter.com/jsngr/status/1287026808429383680?s=20
  - https://stripe.com/
- 產生以及更新圖形
  - https://twitter.com/plotlygraphs/status/1286688715167936512
- 其他
  - https://github.com/elyase/awesome-gpt3

國立高雄科技大學
National Kaohsiung University of Science and Technology

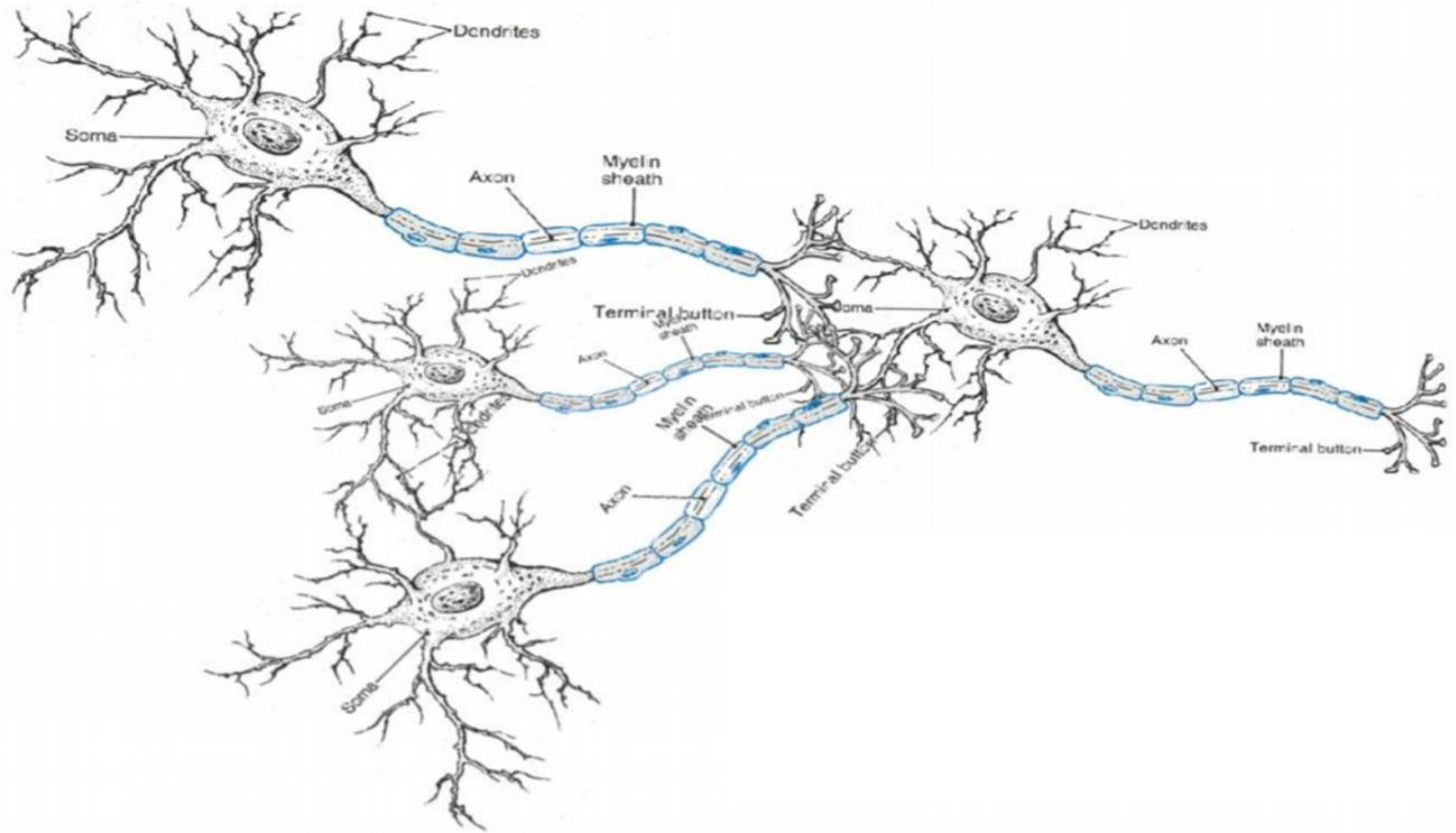
Visual 20 Question Games

Zhang, Zhao & Yu SIGDIAL 2018.

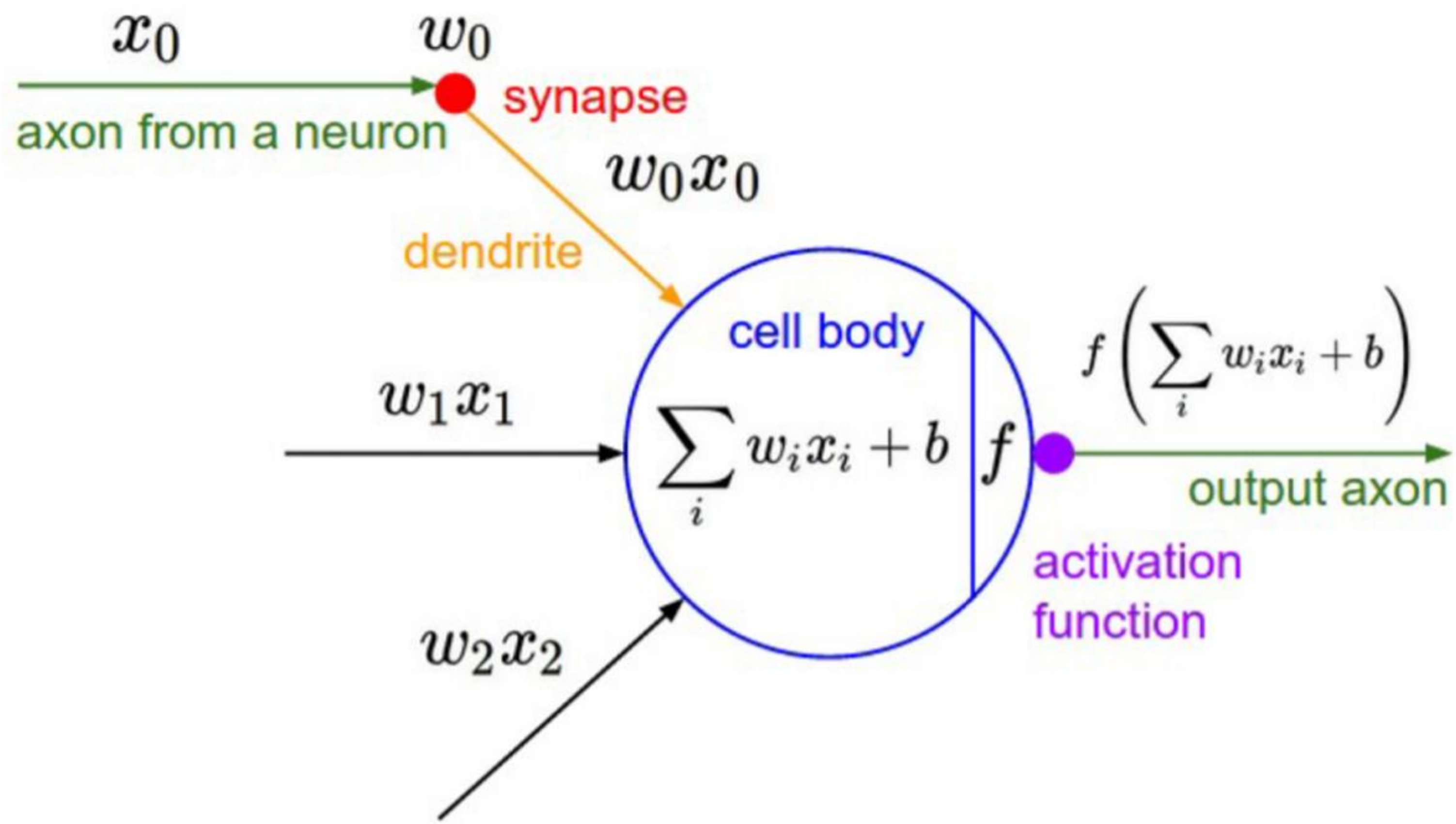

A plate on the table with fried and raw vegetable.
Are there any carrots?
Yes, there are carrots.
Is it the fourth image on the second row?
Correct!

Neural Network

人類的神經網路

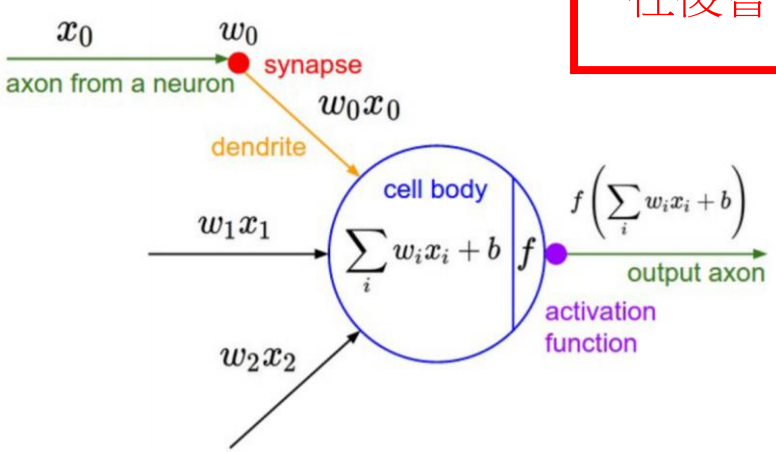

Dendrites
Soma
Axon
Myelin sheath
Dendrites
Terminal button
Soma
Axon
Myelin sheath
Terminal button
Axon
Myelin sheath
Terminal button
Axon
Soma

人工神經網路 - 單一神經元

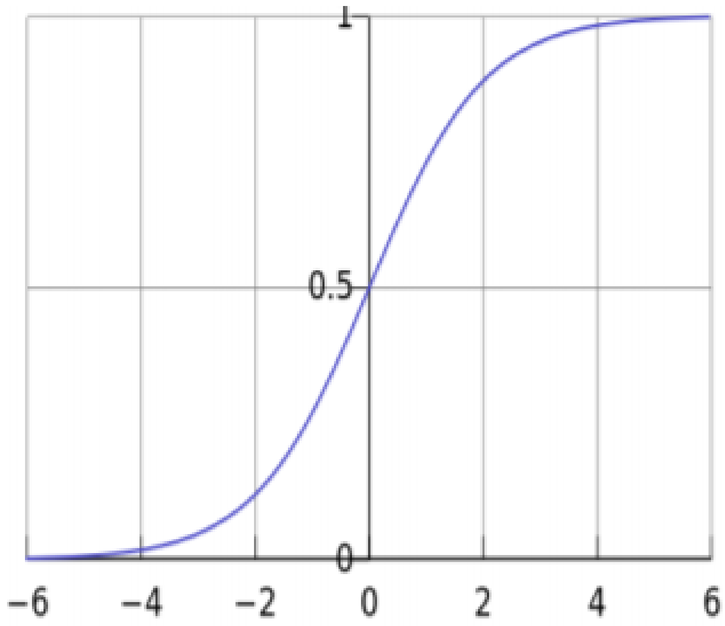


單一神經元的運算

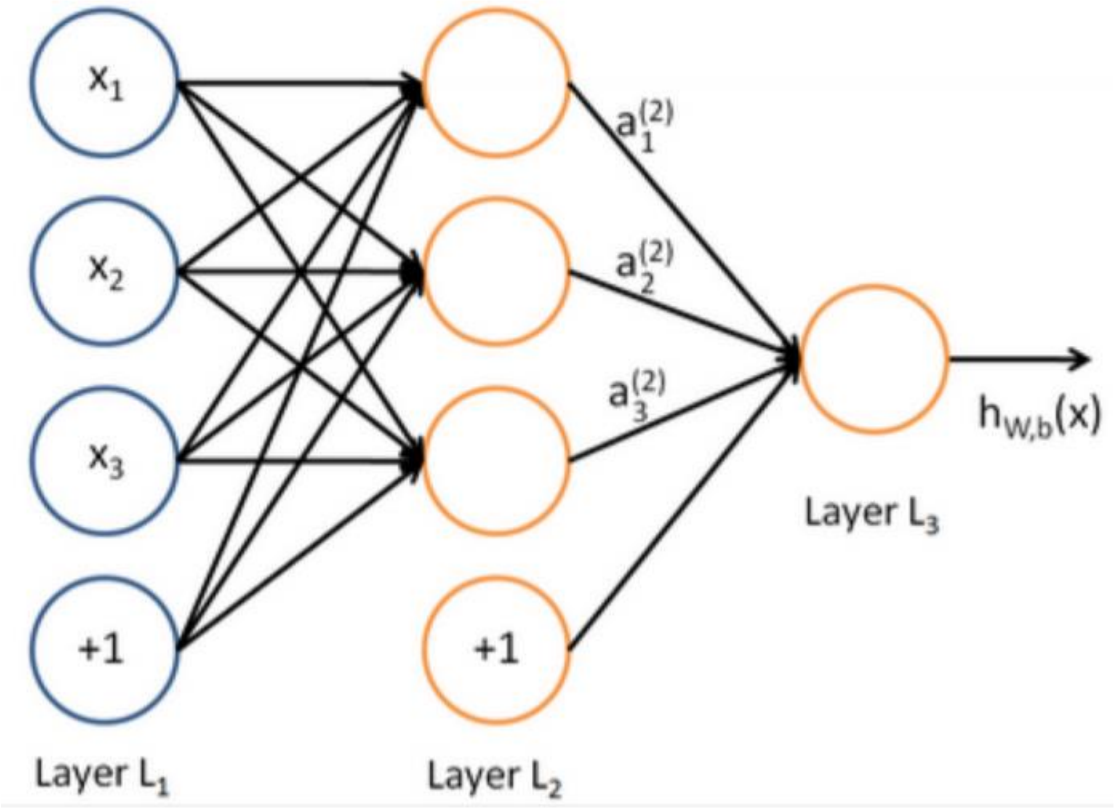
- 假設我們使用Sigmoid當成激活函數
- $z = w^T x + b$
- $f(z) = \frac{1}{1-\exp(-z)}$



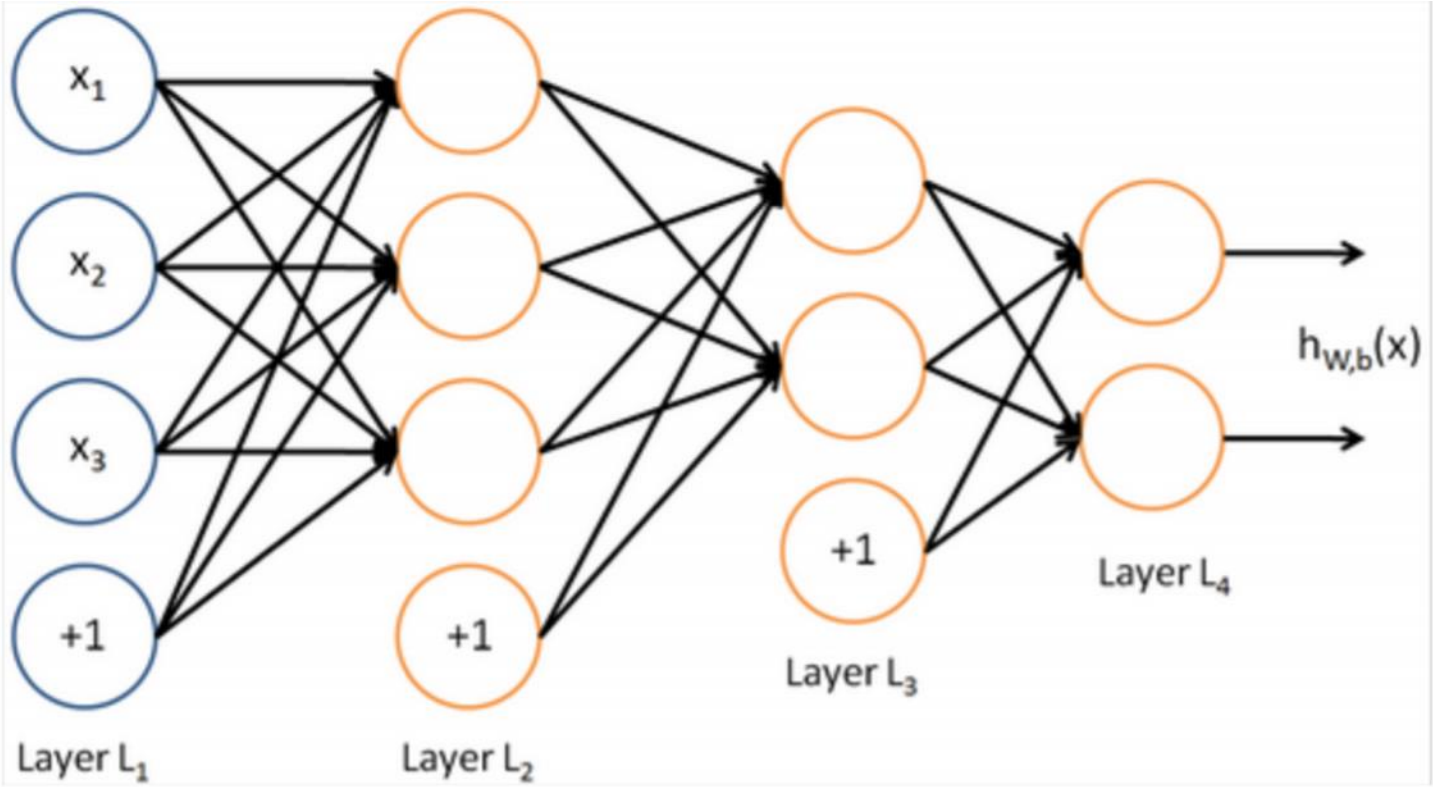
往後會介紹更多不同激活函數



單層人工神經網路



多層人工神經網路



矩陣表示法

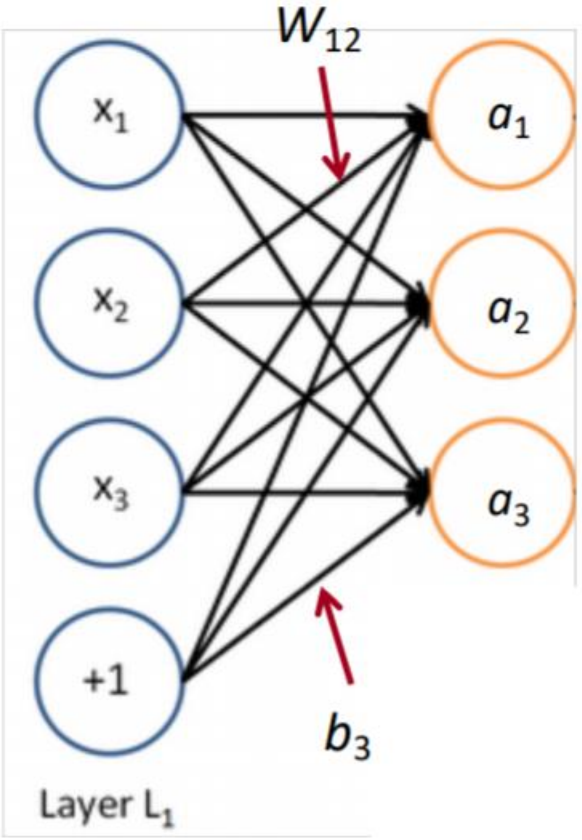
- $a_1 = f(W_{11}x_1 + W_{12}x_2 + W_{13}x_3 + b_1)$

- Matrix Notation

  z= $Wx + b$

  a = f(z)

- Activation function is element-wise
- $f([z_1, z_2, z_3]) = [f(z_1), f(z_2), f(z_3)]$

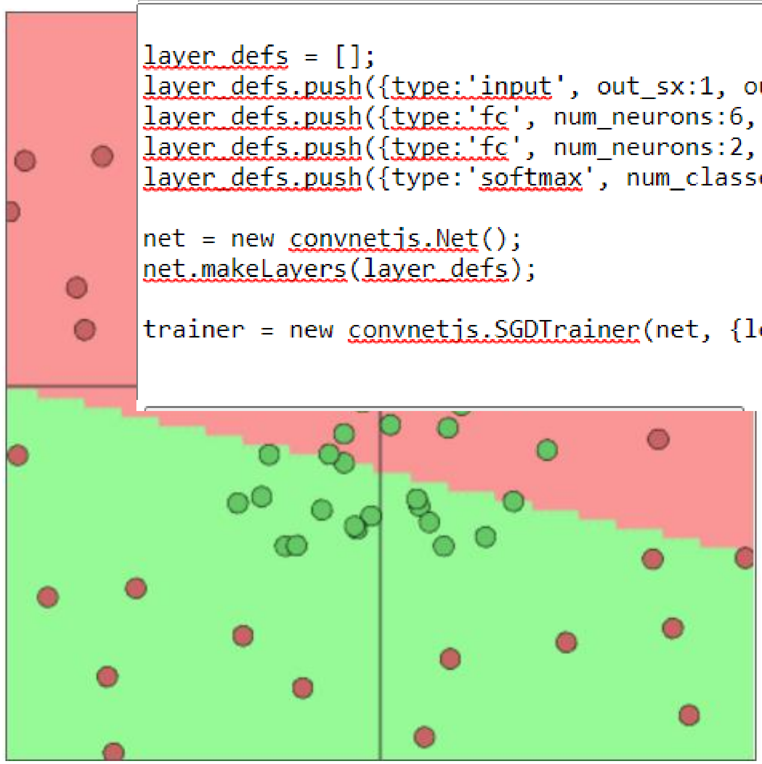


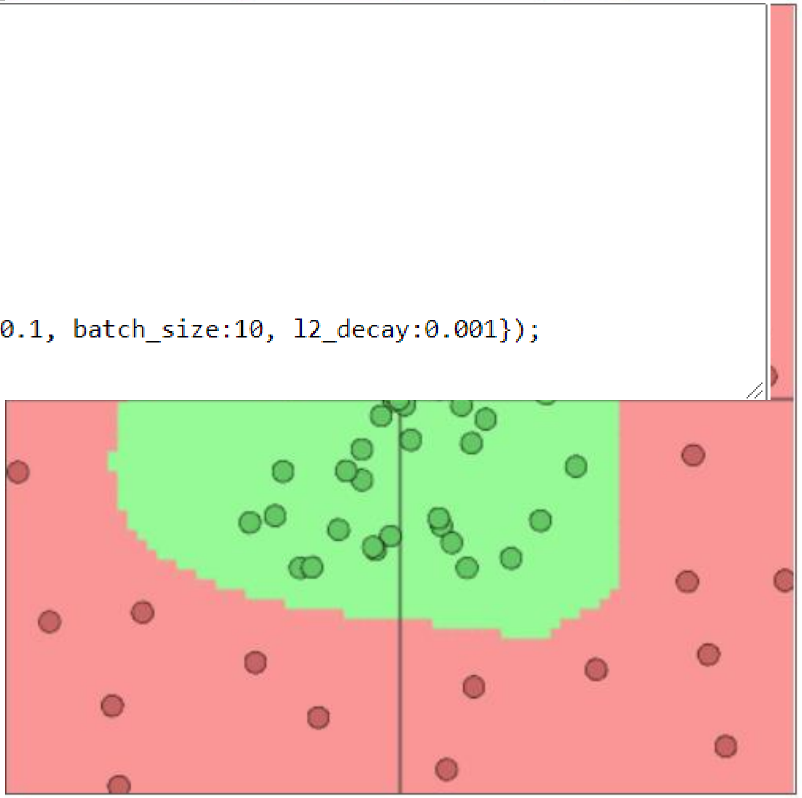
為甚麼要使用非線性的激活函數?

```
layer_defs = [];
layer_defs.push({type:'input', out_sx:1, out_sy:1, out_depth:2});
layer_defs.push({type:'fc', num_neurons:6, activation: 'sigmoid'});
layer_defs.push({type:'fc', num_neurons:2, activation: 'relu'});
layer_defs.push({type:'softmax', num_classes:2});

net = new convnetjs.Net();
net.makeLayers(layer_defs);

trainer = new convnetjs.SGDTrainer(net, {learning_rate:0.01, momentum:0.1, batch_size:10, l2_decay:0.001});
```

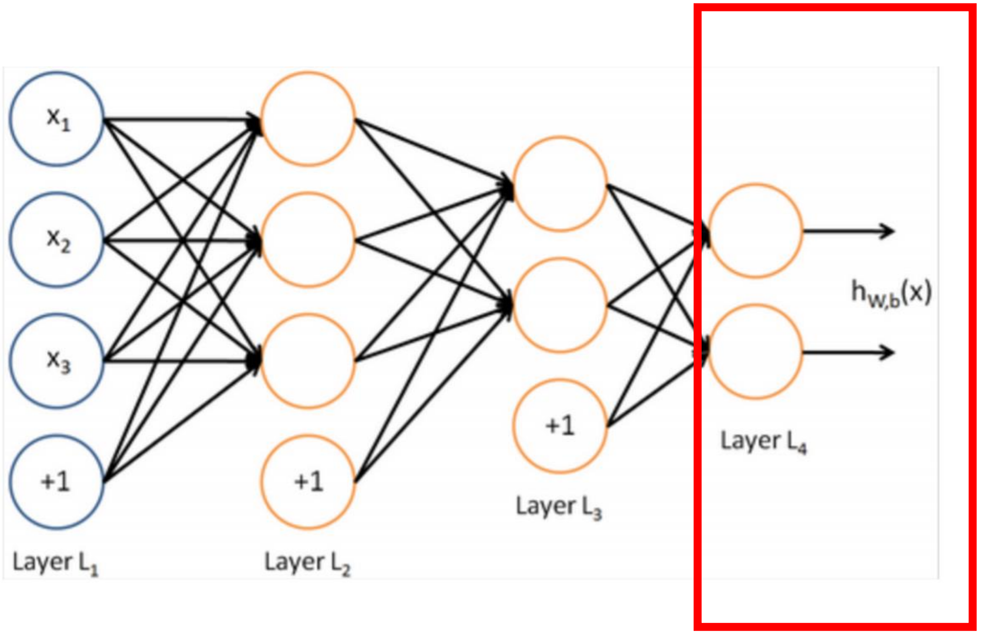




- https://cs.stanford.edu/people/karpathy/convnetjs/demo/classify2d.html

Softmax 函數

- 通常放在輸出
- 介於 0 ~ 1 之間
- $p(y|x) = \frac{\exp(W_y x)}{\sum_{c=1}^{C} \exp(W_c x)}$



Cross-Entropy Loss

- 從資訊理論推倒而來
- 假設有一個One-Hot Vector [0,0,…,1]
- 假設P = True Probability, q = Predicted Probability
- Cross-Entropy 如下:
- $H(p,q) = -\sum_{c=1}^{C} p(c) \log q(c)$
- 由於我們採用One-Hot Encoding, 所以可以將Cross-Entropy重新整理成下列公式:
- $-\log\big(p(y|x)\big) = -\log\left(\frac{\exp(fy)}{\sum_{c=1}^{C} \exp(fc)}\right)$

Softmax with Cross-Entropy Loss

- 目標: 最大化正確類別y的機率
- 然而: 最大化Log 機率等於最小化負的Log機率
- 因此: 公式如下
- $-\log\left(p(y|x)\right) = -\log\left(\frac{\exp\left(W_y x\right)}{\sum_{c=1}^{C} \exp(W_c x)}\right)$

Full Dataset Classification

- 計算整個資料集損失，我們會將Cross-Entropy Loss的值取平均，所以公式如下:
- $J(\theta) = \frac{1}{N}\sum_{i=1}^{N} -\log\left(\frac{\exp(f_{yi})}{\sum_{c=1}^{C}\exp(f_c)}\right)$
- $f_y = W_y x = \sum_{i=1}^{d} W_{yi} x_i$
- 通常會用小Batch而不是把整個資料集丟進去訓練

Stochastic Gradient Decent

- Update Equation:
- $\theta^{new} = \theta^{old} - \alpha \boxed{\nabla_\theta J(\theta)}$, where α is learning rate.

如何計算?

- 手動計算
- 反向傳播法

微分複習

基本微分

- Given R → R function $f(x) = 3x^2$, what is its slope / gradient?

➢ $\frac{df}{dx} = 6x$

- Given $R^n \to R$ function, f(x) = $f(x_1, x_2, x_3) = x_1 + x_2^2 + x_3^3$what is its gradient?

➢ $\frac{\partial f}{\partial x} = [1, 2x_2, 3x_3^2]$

- $R^n \to R$ 函數的微分公式:

➢$\frac{\partial f}{\partial x} = \left[\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \dots, \frac{\partial f}{\partial x_n}\right]$

Jacobian Matrix : Vectorized Gradients

- 目前所學的知識: 對單一參數微分
- 然而: 對單一參數微分較無效率
- 假設我們有一個函數: $R^n \to R^m,\ \ f(x)\ = [f_1(x_1 \dots x_n), f_2(x_1 \dots x_n), \dots, f_m(x_1 \dots x_n)]$
- Jacobian Matrix 如下:
- $\frac{\partial f}{\partial x} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial n} \end{bmatrix}, R^{m*n}$

連鎖律

- Jacobian Matrix 的點積
- 假設有一個函數 $h(x) = f(g(x))$，其微分如下:

➢ $J_{h(x)} = J_{f(g(x))} \cdot J_{g(x)}$

傳統連鎖律的運算

- Suppose we have $f\left(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}\right) \rightarrow 3x_1 + x_2^2$ and $g\left(\begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix}\right) = (\begin{pmatrix} y_1 + 2y_2 + 3y_3 \\ y_1 y_2 y_3 \end{pmatrix})$, what is the gradients of h = f ∘ g ?

➢ h$\left(\begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix}\right) = f\left(\begin{pmatrix} y_1 + 2y_2 + 3y_3 \\ y_1 y_2 y_3 \end{pmatrix}\right) = 3(y_1 + 2y_2 + 3y_3) + (y_1 y_2 y_3)^2$

➢ $\frac{\partial h}{\partial y_1}(y) = 3 + 2y_1 y_2^2 y_3^2, \frac{\partial h}{\partial y_2}(y) = 6 + 2y_2 y_1^2 y_3^2, \frac{\partial h}{\partial y_3}(y) = 9 + 2y_3 y_2^2 y_1^2$

- so the gradients are

➢ $\nabla_y h = \begin{bmatrix} 3 + 2y_1 y_2^2 y_3^2 \\ 6 + 2y_2 y_1^2 y_3^2 \\ 9 + 2y_3 y_2^2 y_1^2 \end{bmatrix}$

Multiplication of Jacobian Matrix Chain Rule

- The same example as above let's calculate chain rule with multiplication of Jacobian Matrix.

➢ $\nabla_x f = \begin{pmatrix} 3 & 2x_2 \end{pmatrix}, \nabla_y g = \begin{pmatrix} 1 & 2 & 3 \\ y_2 y_3 & y_1 y_3 & y_1 y_2 \end{pmatrix}$

➢ $\nabla_{g(y)} f \left(\begin{pmatrix} y_1 + 2y_2 + 3y_3 \\ y_1 y_2 y_3 \end{pmatrix} \right) = \begin{pmatrix} 3 & 2y_1 y_2 y_3 \end{pmatrix}$

➢ $\nabla_y h = \nabla_{g(y)} f \cdot \nabla_y g = \begin{bmatrix} 3 & 2y_1 y_2 y_3 \end{bmatrix} \cdot \begin{pmatrix} 1 & 2 & 3 \\ y_2 y_3 & y_1 y_3 & y_1 y_2 \end{pmatrix}$

Useful Identities for Neural Network

Matrix Times Column Vector w.r.t Column Vector (Jacobian Matrix)

- z = Wx, what is $\frac{\partial z}{\partial x}$? W ϵ R^{n*m}, x ϵ R^{m}
- z = $\begin{bmatrix} W_{11}x_1 + W_{12}x_2 + \cdots + W_{1m}x_m \\ \vdots \\ W_{n1}x_1 + W_{n2}x_2 + \cdots + W_{nm}x_m \end{bmatrix}$
- $\frac{\partial z}{\partial x} = \begin{bmatrix} W_{11} & \cdots & W_{1m} \\ \vdots & \ddots & \vdots \\ W_{n1} & \cdots & W_{nm} \end{bmatrix}$ = W – (1)

Row Vector Times Matrix w.r.t Row Vector

- z = xW, what is $\frac{\partial z}{\partial x}$? W $\epsilon\ R^{n*m}$, x $\epsilon\ R^{1*n}$

➢ z = $[W_{11}x_1 + W_{12}x_2 + \cdots + W_{1m}x_m, \ldots, W_{n1}x_1 + W_{n2}x_2 + \cdots + W_{nm}x_m]$

- $\frac{\partial z}{\partial x} = \begin{bmatrix} W_{11} & \ldots & W_{n1} \\ \vdots & \ddots & \vdots \\ W_{1m} & \ldots & W_{nm} \end{bmatrix}$ = W^T – (2)

A Vector w.r.t Itself

- z = x, what is $\frac{\partial z}{\partial x}$? x $\epsilon\ R^m$

➢ z = $\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}$

➢ $\frac{\partial z}{\partial x}$ = $\begin{bmatrix} 10 \cdots 0 \\ 0\ 1 \cdots 0 \\ \vdots \\ 00 \cdots 1 \end{bmatrix}$ = I

- Thus, $\frac{\partial z}{\partial x}$ = I, which is an identity matrix. – (3)

Elementwise Function

- z = f(x), what is $\frac{\partial z}{\partial x}$?
  - z = $\begin{bmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_m) \end{bmatrix}$
  - $\frac{\partial z}{\partial x}$ = $\begin{bmatrix} f'(x_1), \dots, 0 \\ 0, f'(x_2), \dots \\ \vdots \\ 0, \dots, f'(x_m) \end{bmatrix}$
- Thus, $\frac{\partial z}{\partial x}$ = diag($f'(x)$) – (4)

Matrix Times Column Vector w.r.t Matrix

- z = Wx, $\delta = \frac{\partial J}{\partial z}$, what is $\frac{\partial J}{\partial W}$? W ϵ R^{n*m}, x ϵ R^m
  - z = $\begin{bmatrix} x_1W_{11} + x_2W_{12} + \cdots + x_mW_{1m} \\ \vdots \\ x_1W_{n1} + x_2W_{n2} + \cdots + x_mW_{nm} \end{bmatrix}$
  - $\frac{\partial J}{\partial W}$ = $\frac{\partial J}{\partial z}$ $*$ $\frac{\partial z}{\partial w}$ =δ $*$ $\frac{\partial z}{\partial w}$ (根據連鎖律)
  - $\frac{\partial J}{\partial w_{11}} = \frac{\partial J}{\partial z} \frac{\partial z}{\partial w_{11}}$ = $\begin{bmatrix} \frac{\partial J}{\partial z_1} \\ \vdots \\ \frac{\partial J}{\partial z_n} \end{bmatrix} \begin{bmatrix} x_1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$ = $\left[\frac{\partial J}{\partial z_1}, \ldots, \frac{\partial J}{\partial z_n}\right] \begin{bmatrix} x_1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \delta_1 * x_1$

    n - 1 個0
- ....

Matrix Times Column Vector w.r.t Matrix

- Therefore, $\frac{\partial J}{\partial W}$ = $\begin{bmatrix} \delta_1 * x_1 , \delta_1 * x_2, \ldots , \delta_1 * x_m \\ \vdots \\ \delta_n * x_1 , \delta_n * x_2, \ldots , \delta_n * x_m \end{bmatrix}$ = $\begin{bmatrix} \delta_1 \\ \vdots \\ \delta_n \end{bmatrix}$ $[x_1, \ldots, x_m]$ =$\delta\ x^T$

Derivative of Cross-Entropy Loss w.r.t x

- J = $-\log(p(y|x)) = -\log\left(\frac{\exp(W_y x)}{\sum_{c=1}^{C}\exp(W_c x)}\right)$, what is $\frac{\partial J}{\partial x}$?

➢ J = $-W_y x + \log\left(\sum_{c=1}^{C}\exp(W_c x)\right)$

➢ $\frac{\partial J}{\partial x} = \boxed{-W_y} + \frac{\exp(W_y x)}{\sum_{c=1}^{C}\exp(W_c x)} W_y$

Identity 1

Derivative of Cross-Entropy Loss

- J = $-\log\left(\frac{\exp(z_i)}{\sum_{c=1}^{C}\exp(z_c)}\right)$, what is $\frac{\partial J}{\partial z_i}$?

➢ J = -z_i+ $\log\left(\sum_{c=1}^{C}\exp(z_c)\right)$

➢ $\frac{\partial J}{\partial z_i}$ = -1 + $\boxed{\frac{\exp(z_i)}{\sum_{c=1}^{C}\exp(z_c)}}$

➢ = -1 + softmax(z_i)

Derivative of Sigmoid Function

- f(z) = $\frac{1}{1-\exp(-z)}$, what is $\frac{\partial f}{\partial z}$?

➢ $\frac{\partial f}{\partial z} = \frac{-\exp(-z)}{[1-\exp(-z)]^2}$

$= \frac{-\exp(-z)+1-1}{[1-\exp(-z)]^2}$

$= \frac{1}{1-\exp(-z)} * \boxed{\frac{-\exp(-z)+1-1}{1-\exp(-z)}}$

$= \frac{1}{1-\exp(-z)} * \left[1 + \boxed{\left(\frac{-1}{1-\exp(-z)}\right)}\right]$

$= \sigma(z) * [1 + \sigma(z)]$

Derivative of ReLU Function

- ReLU(x) = max(0,x), 求 ReLU 的微分

➢ $ReLU'(x) = \begin{cases} 1, if x > 0 \\ 0, otherwise \end{cases}$

Derivative of Tanh Function

- Tanh(z) = $\frac{\exp(z)-\exp(-z)}{\exp(z)+\exp(-z)}$, 求Tanh的微分

➢ $\text{Tanh}'(z) = \frac{[\exp(z)+\exp(-z)]*[\exp(z)+\exp(-z)] - [\exp(z)-\exp(-z)]*[\exp(z)-\exp(-z)]}{[\exp(z)+\exp(-z)]^2}$

$= \frac{[\exp(z)+\exp(-z)]^2 - [\exp(z)-\exp(-z)]^2}{[\exp(z)+\exp(-z)]^2}$

$= 1 - \text{Tanh}^2(z)$

References

- 斎藤康毅, Deep Learning 2: 用Python進行自然語言處理的基礎理論實作.
- Manning et al., CS224n Natural Language Processing with Deep Learning, Stanford University.
- Guillaume Genthial, Review of differential calculus theory.
- Kevin Clark, Computing Neural Network Gradients.