Deep Neural Network & Recommendation Engine on TensorFlow

Gary Chen

Outline

- Machine Learning Framework
- TensorFlow Fundamentals
- Deep Neural Network(DNN) Tips
- Recommedation Engine in Matrix Factorization
- Matrix Factorization with DNN

Machine Learning 框架

- Machine Learning Framework
- TensorFlow Fundamentals
- Deep Neural Network(DNN) Tips
- Recommedation Engine in Matrix Factorization
- Matrix Factorization with DNN

先來看則漫畫~

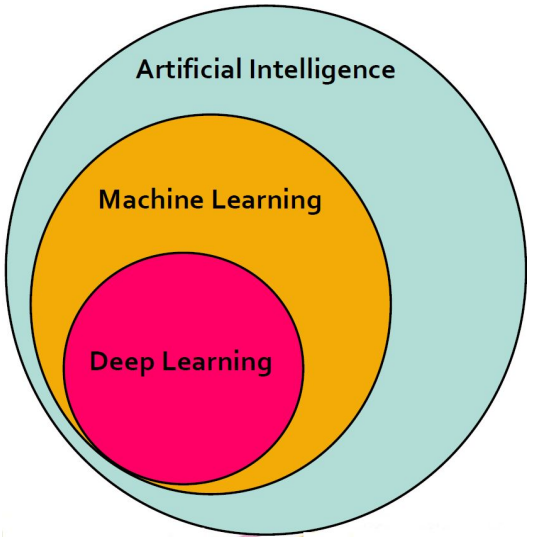
用IF ELSE堆積而成的AI

↓

Rule Base

Machine Learning in Artifical Intelligence

- AI只是達成人工智能的一種過程
  - 成千上萬的rule base code可以是一個達成AI的手段
- Machine Learning
  - 從資料中學習rules
  - 在一堆function set中決定一個最好的function => Model



You Write a programe for learning

Machine Learning Framework

- Machine Learning 3 Steps

Step1 — **Define a set of functions**

Step2 — **Evaluate and search**

Step3 — **Pick the best function**

如何知道Function好壞？

Functions Set

Model

f1, f2...

f1()= ”夏雨喬”

f1()= ”宋雲樺”

Better

f2()= ”陳國坤”

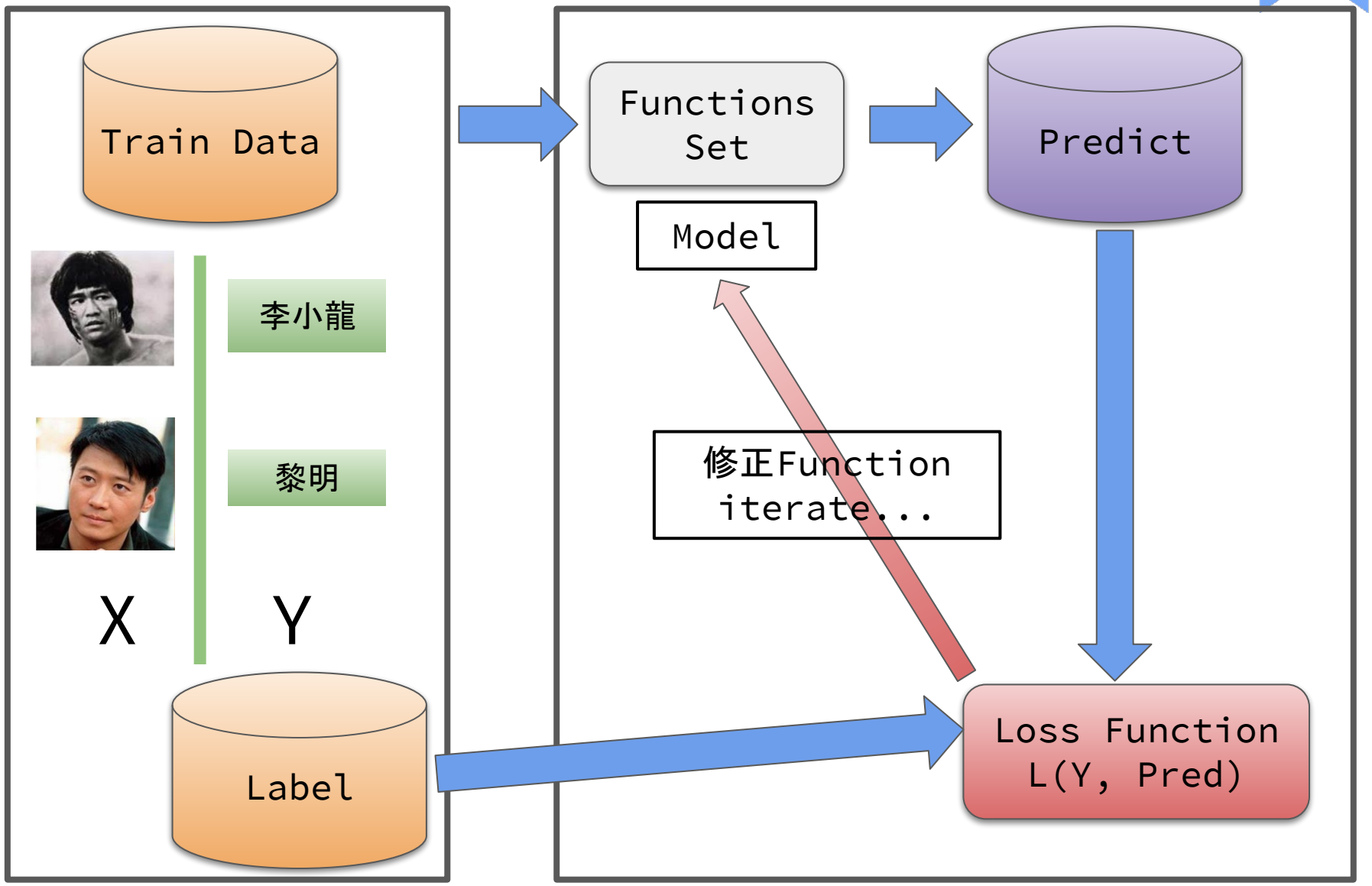
f2()= ”林國斌”

Train Data

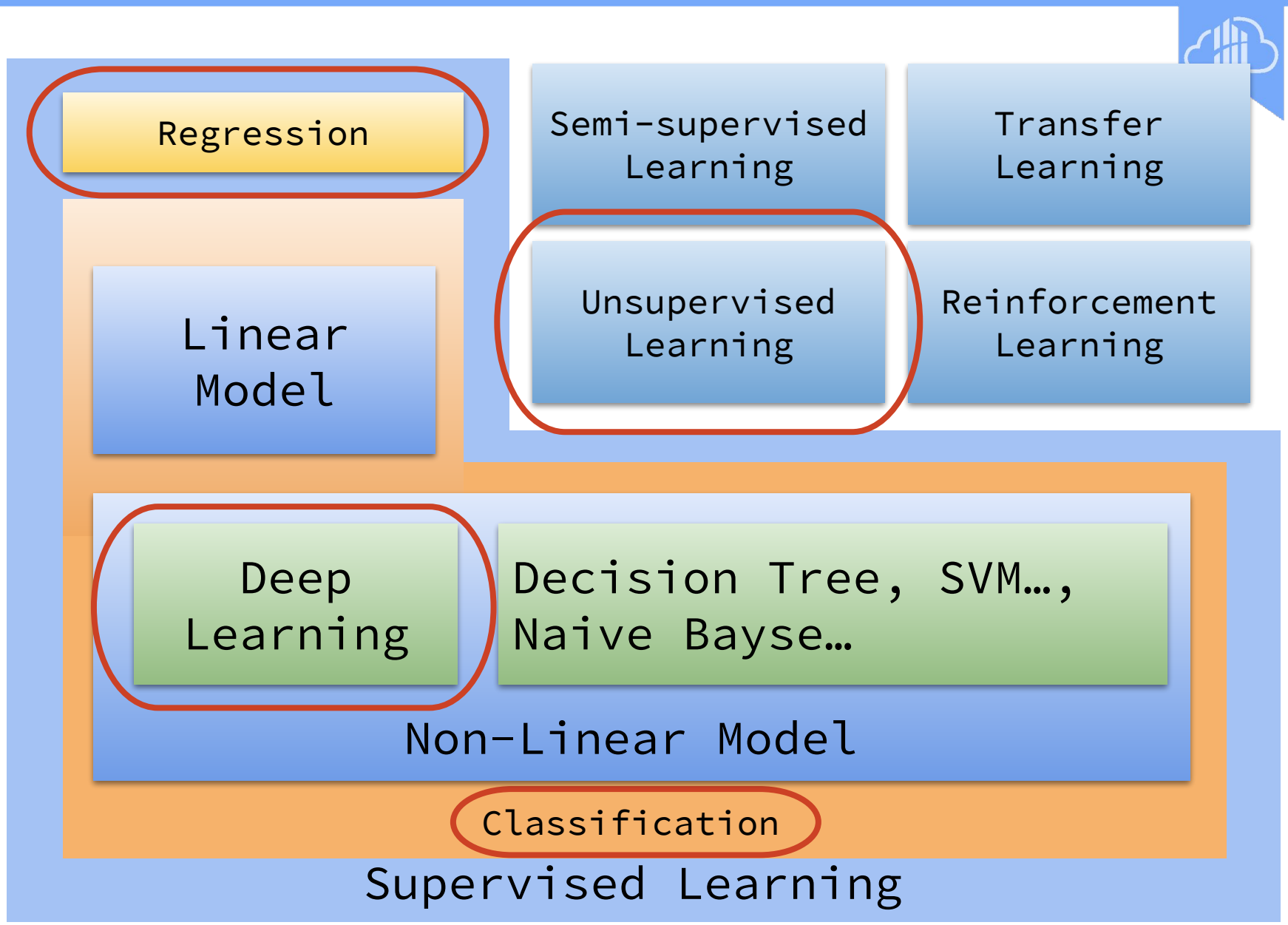
Input				
Answer	夏雨喬	宋雲樺	李小龍	黎明

Machine Learning Framework

❑ Machine Learning 3 Steps in Detail



Machine Learning 的種類



Unsupervised Learning

- Unsupervised Learning
  - Def: 給一堆沒有label的資料，自己去找答案

Unlabelled Documents → **Clustering**

- 沒有Label就沒有Loss Function，要如何最佳化?

Functions Set ?

Evaluate?

Best Function?

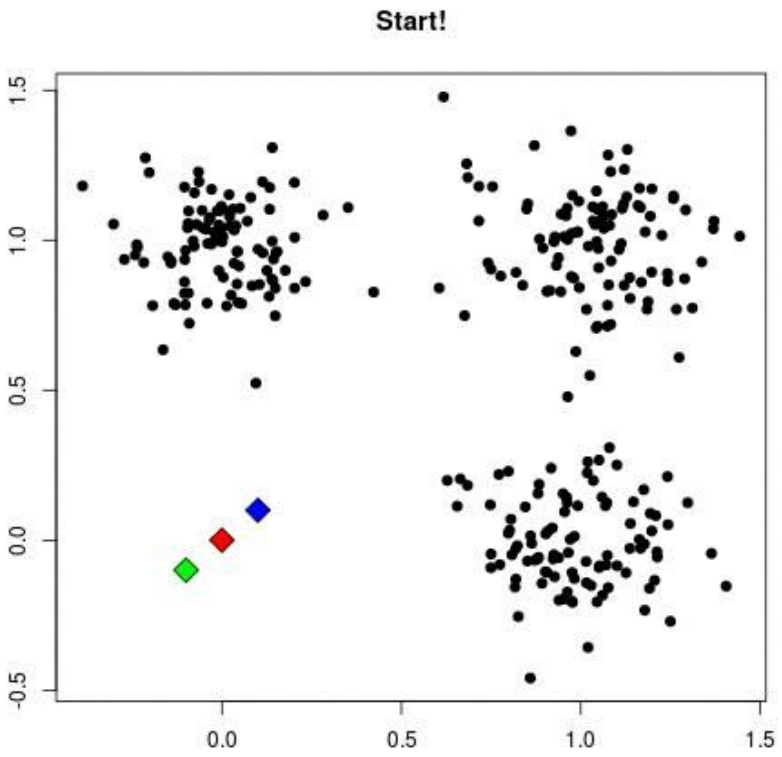
- Unsupervised Learning
  - 給一堆沒有label的資料，自己去找答案
  - 可是**沒有Label就沒有Loss Function**，如何最佳化Model?

K-Means Clustering

- step1: Functions Set (objective function)

$$J = \sum_{i=1}^{k} \sum_{x_j \in S_i} \left(x_j - \mu_i \right)^2$$

- step2: Evaluate
  迭代(Training)直到J最小
- step3: Pick the Best
  實際上會指定迭代次數，直到結果可以接受。



Supervised Learning

- Supervised Learning
  - Def: 有Label的Data中，找一個最好的Model(答案)
  - Regression

    預測Target = 數值(Continous)

    房價預測

    f(坪數, 地理位置...) = Price

    Recommendation

    f(使用者A 商品B) = 對商品的評分

  - Classification

    預測Target = 類別(Discrete)

    手寫圖片辨識

    f() = 圖片數字(0, 1, 2, … 9)

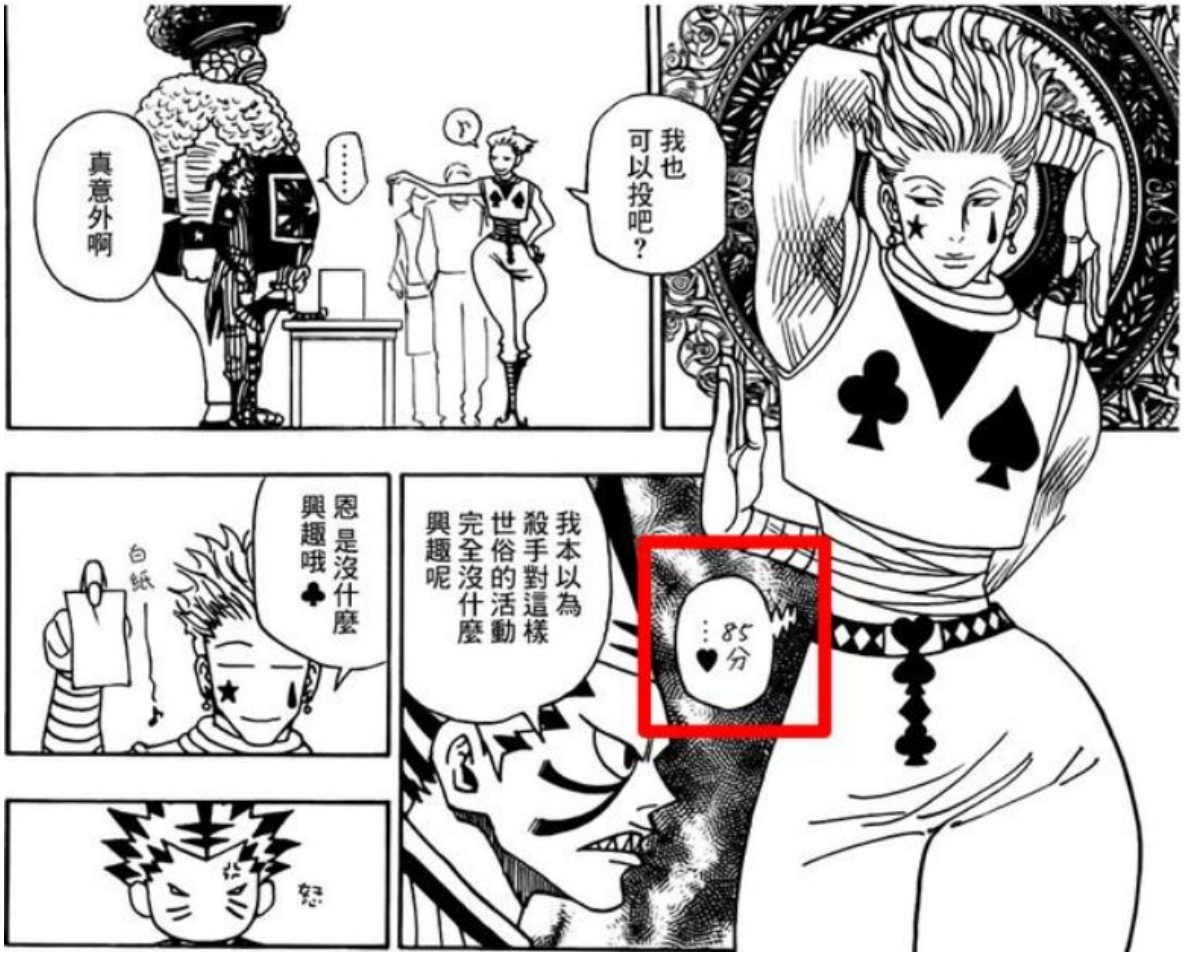
    醫療症狀診斷

    f(發病症狀、年齡、性別...) = 疾病類別

Regression Case Study(Pseudo)

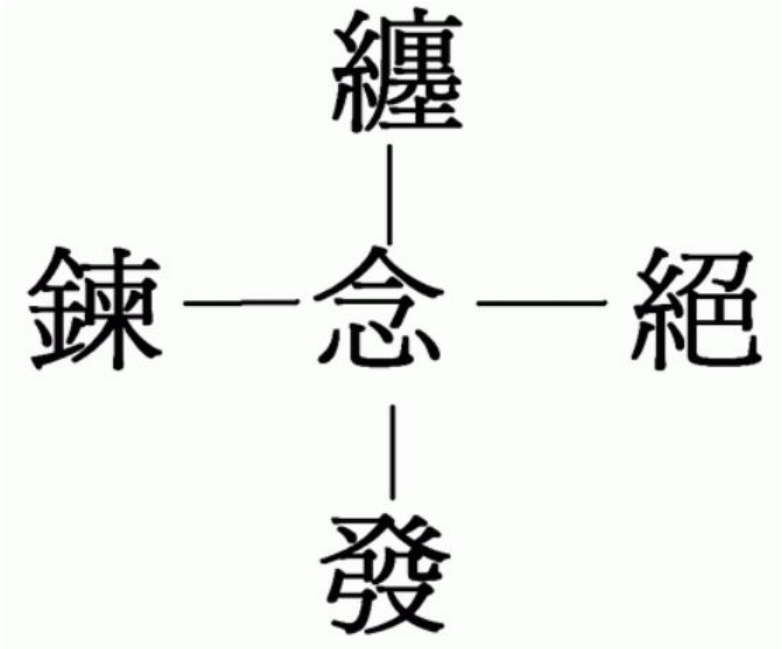
Supervised Learning - Regression

- Regression - Case Study
  HunterXHunter 獵人實力分析
  西索對獵人協會中的尼特羅會長底下的十二地支的評分



Supervised Learning - Regression

- Regression - Case Study
  HunterXHunter 獵人實力分析



知道纏，記住絕，經過錬，達到發

決定一個獵人的實力：

1. 先看念能力氣量
2. 再看系別

Supervised Learning – Regression

- Regression – Case Study
  HunterXHunter 獵人實力分析

Input X: 念能力的氣含量
Output Y: 0 ~ 100分

	X	Y
0	0.019401	0.028230
1	0.077048	0.445230
2	0.083456	0.522365
3	0.121452	1.106286
4	0.129008	1.248226
5	0.504904	19.119641
6	0.585431	25.704695
7	0.646769	31.373290
8	0.710060	37.813880
9	0.734696	40.483402
10	0.898237	60.512242
11	0.995346	74.303532

Supervised Learning - Regression

Step 1: Model(Functions Set)

西索對獵人協會中的**尼特羅會長底下的十二地支**的評分

Input X: 念能力的氣含量
Output Y: 0 ~ 100分

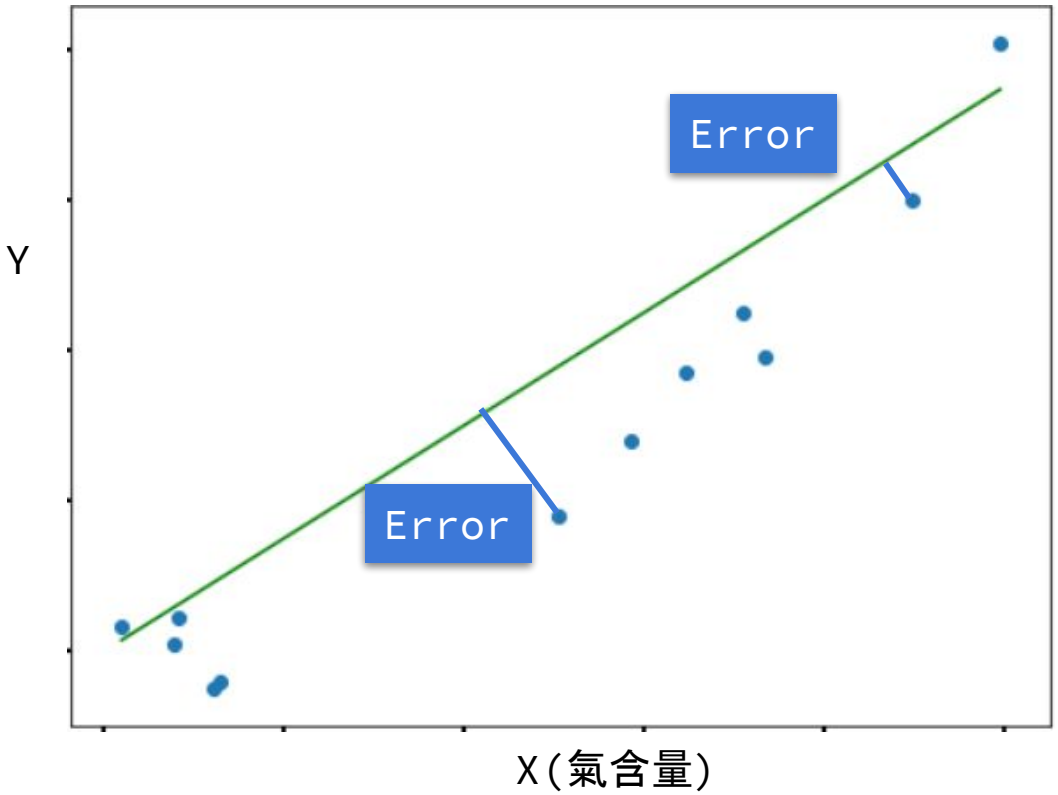
f() = score

Functions Set

$$y = wx + b$$

f1: y = 30X + 0.5
f2: y = 0.9X + 26
f3: y = -1.8X - 0.5

… infinite



Step 2: Evaluate

西索對獵人協會中的**尼特羅會長底下的十二地支**的評分

Input X: 念能力的氣含量
Output Y: 0 ~ 100分

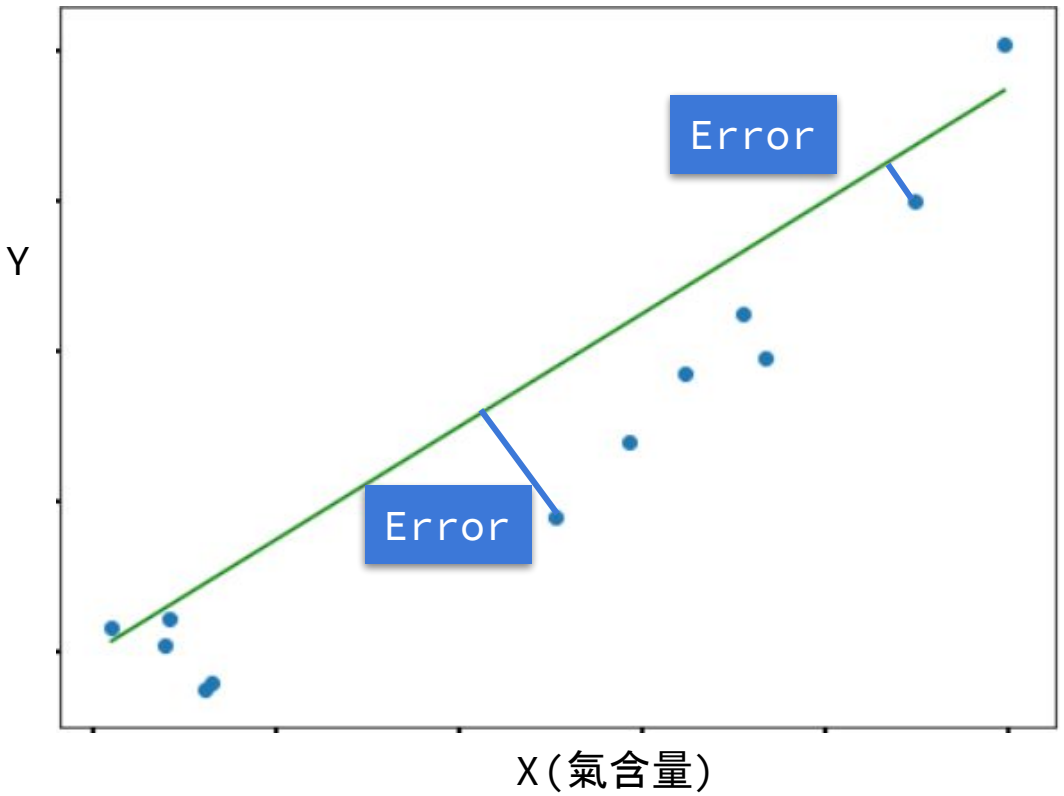
Functions Set

$$y = wx + b$$

Loss function: L

$$L(f) = L(w, b)$$
$$= \sum_{i=1}^{12} (\hat{y}_i - (wx_i + b))^2$$

fn

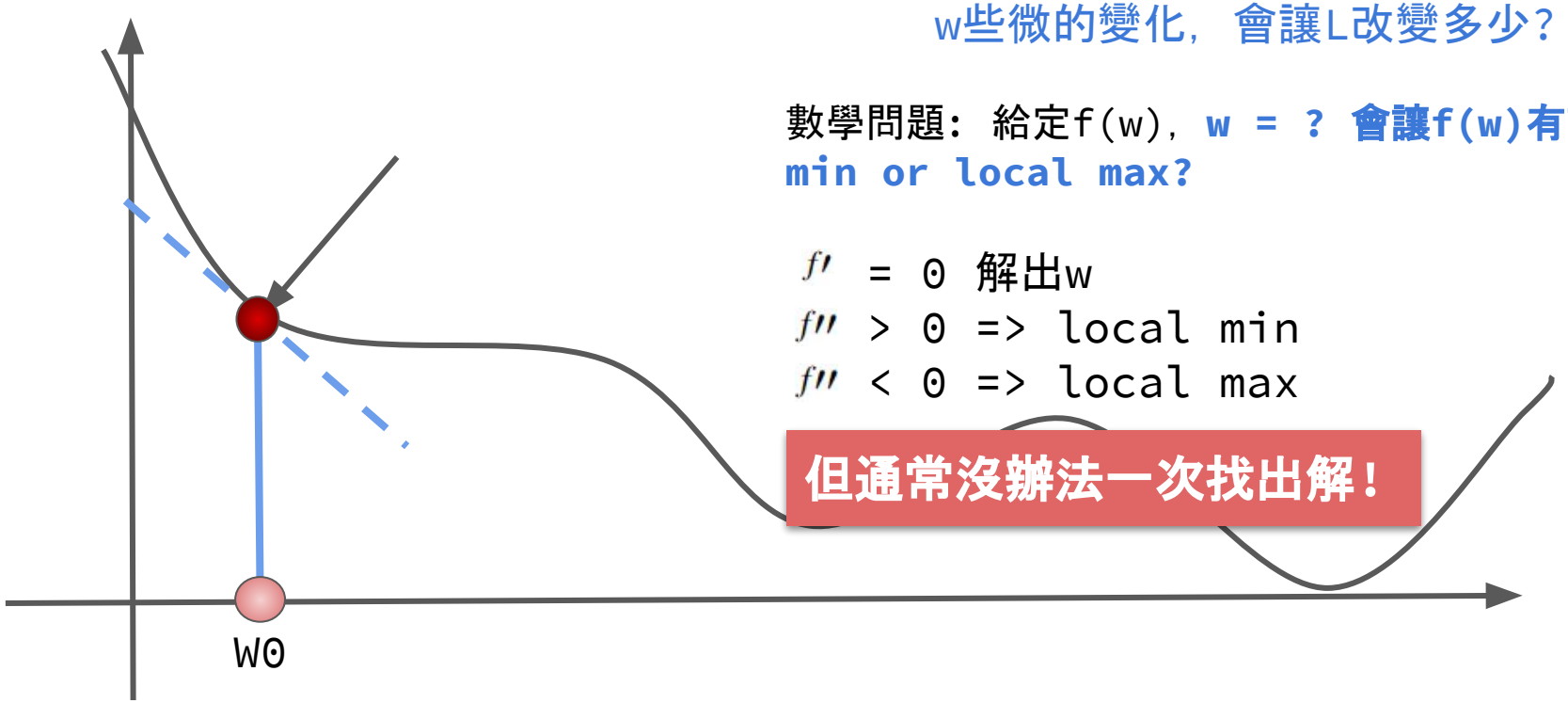


Step 3: Pick Best Function

目標： $\underset{w,b}{\operatorname{argmin}} \sum_{i=1}^{12} (\hat{y}_i - (wx_i + b))^2$ ➡ **最佳化問題！**

微分的定義： $\lim_{\Delta w \to 0} \frac{L(w + \Delta w) - L(w)}{\Delta w}$

w些微的變化，會讓L改變多少？



數學問題：給定f(w)，**w = ? 會讓f(w)有local min or local max?**

f' = 0 解出w

f'' > 0 => local min

f'' < 0 => local max

但通常沒辦法一次找出解！

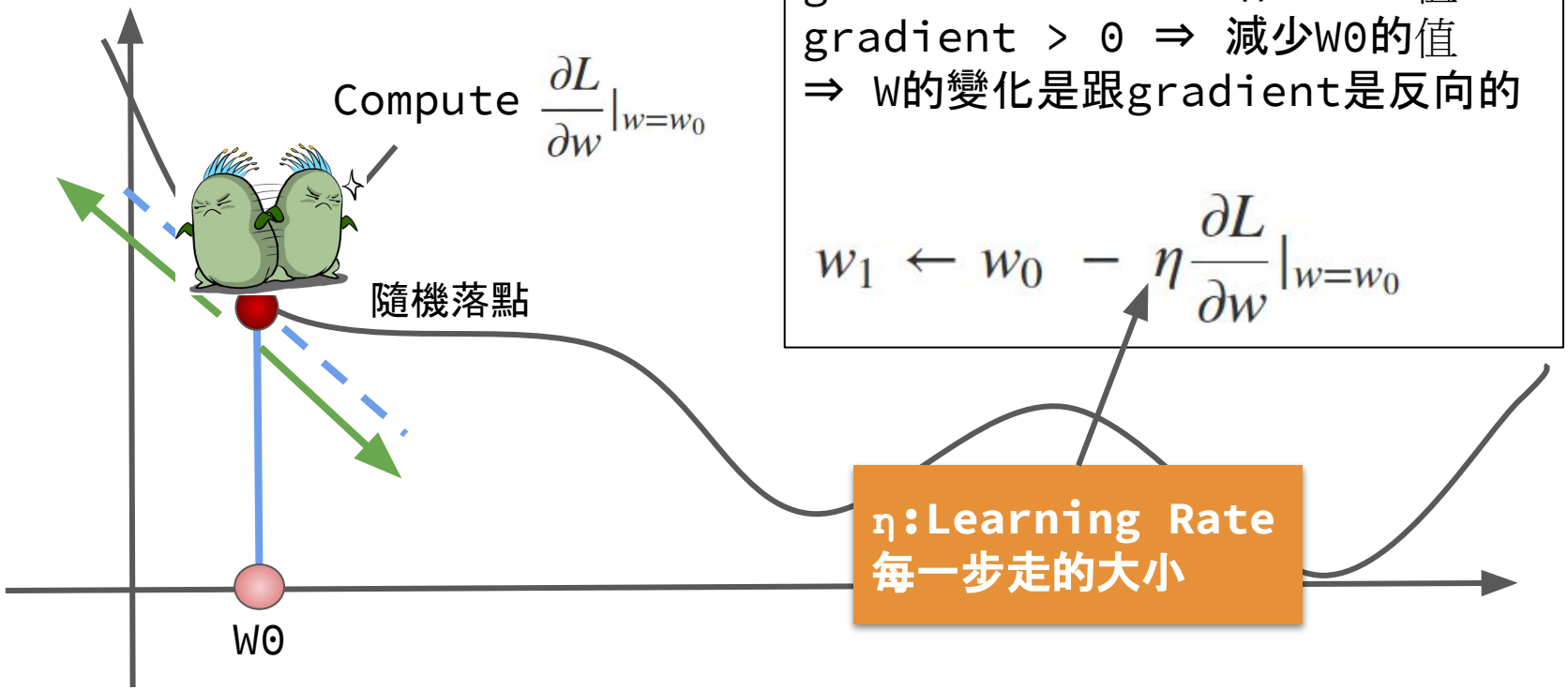
Step 3: Pick Best Function

目標： $\underset{w,b}{\operatorname{argmin}} \sum_{i=1}^{12} (\hat{y}_i - (wx_i + b))^2$ ⇒ **Gradient Descent**

更新W為例

In this case
gradient < 0 ⇒ 增加W0的值
gradient > 0 ⇒ 減少W0的值
⇒ W的變化是跟gradient是反向的

$$w_1 \leftarrow w_0 - \eta \frac{\partial L}{\partial w}|_{w=w_0}$$



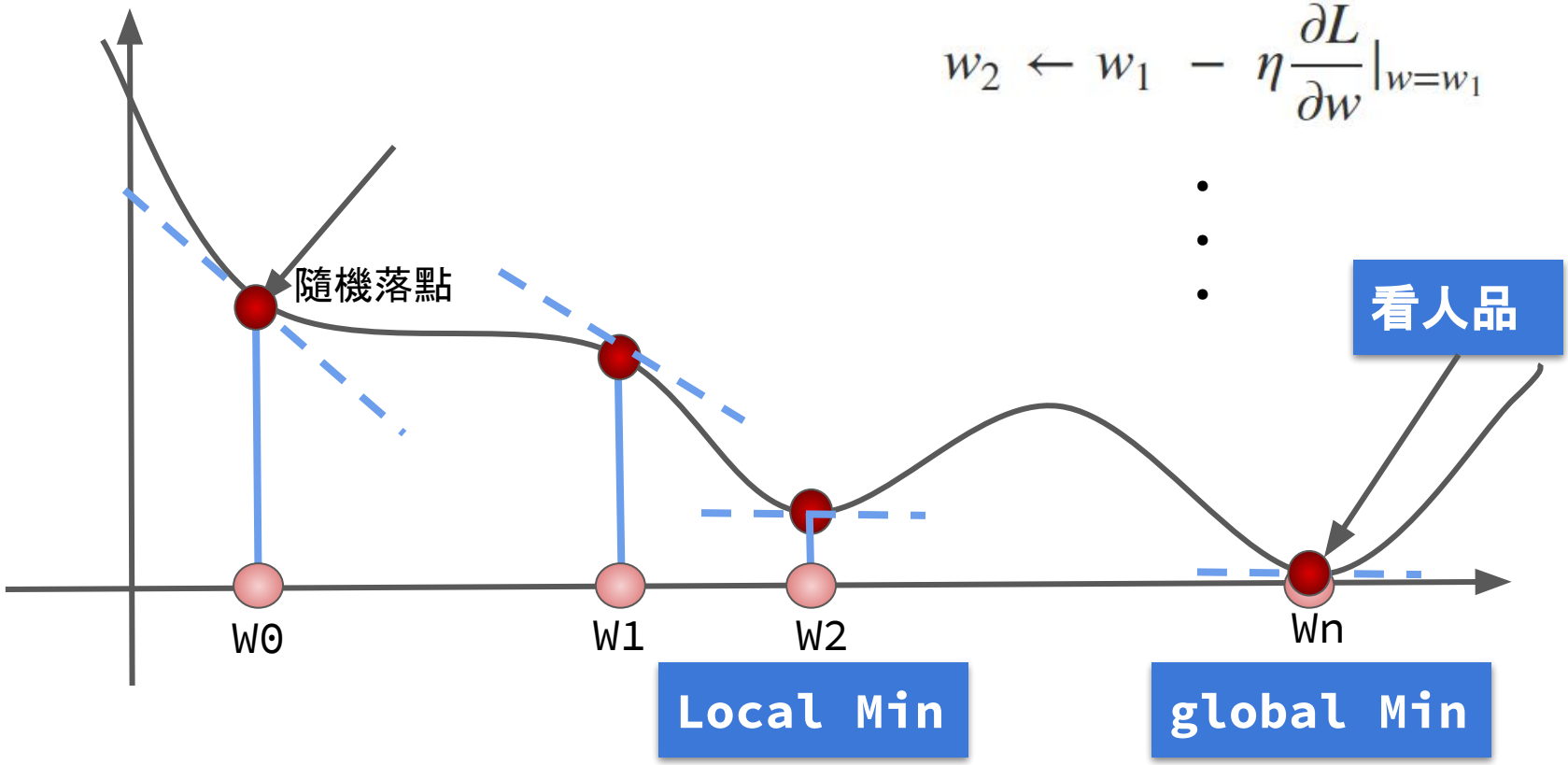
Step 3: Gradient Descent

目標： $\underset{w,b}{\operatorname{argmin}} \sum_{i=1}^{12} (\hat{y}_i - (wx_i + b))^2$

Gradient Descent

$$w_1 \leftarrow w_0 - \eta \frac{\partial L}{\partial w}\Big|_{w=w_0}$$

$$w_2 \leftarrow w_1 - \eta \frac{\partial L}{\partial w}\Big|_{w=w_1}$$



Step 3: Gradient Descent

Chain Rule

$$x,\ u = f_1(x),\ v = f_2(u)$$

$$\frac{dv}{dx} = \frac{dv}{du}\frac{du}{dx}$$

$$L = \sum (\hat{y} - (b + wx))^2$$

$$\frac{\partial L}{\partial w} = \sum 2(\hat{y} - (b + wx))(-x)$$

$$\frac{\partial L}{\partial b} = \sum 2(\hat{y} - (b + wx))(-1)$$

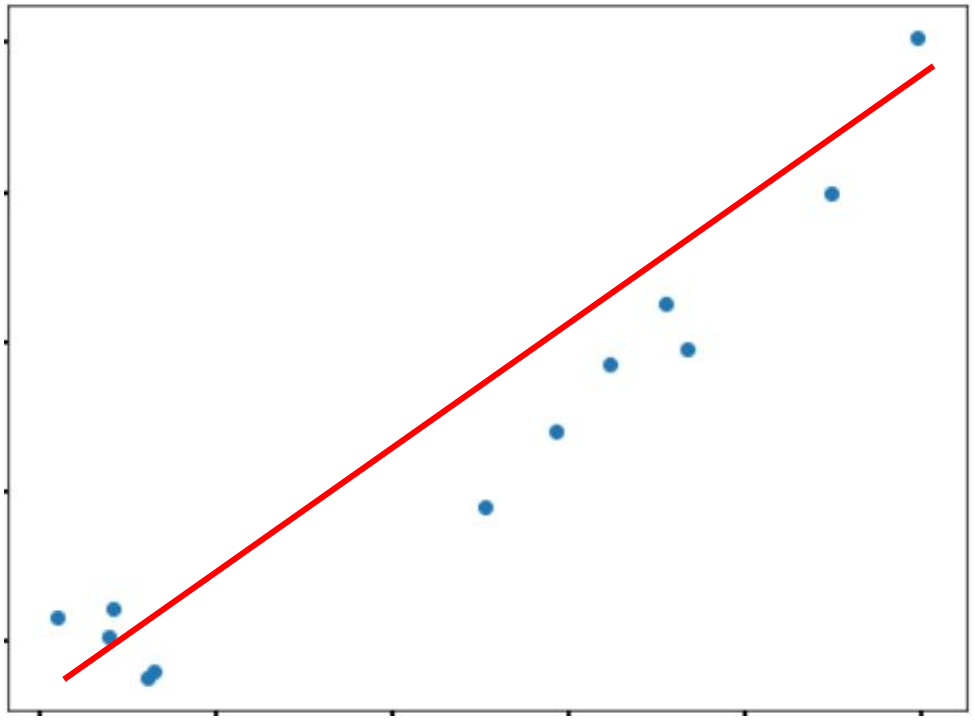
Train Data上與Test Data上的表現

$$y = wx + b$$

假設...

train data error = 31
test data error = 35

w, b再怎麼調整都是線性
⇒ **嘗試引入2次式!**



Supervised Learning - Regression

Reconstruct Model

$$y = w_2 x^2 + w_1 x + b$$

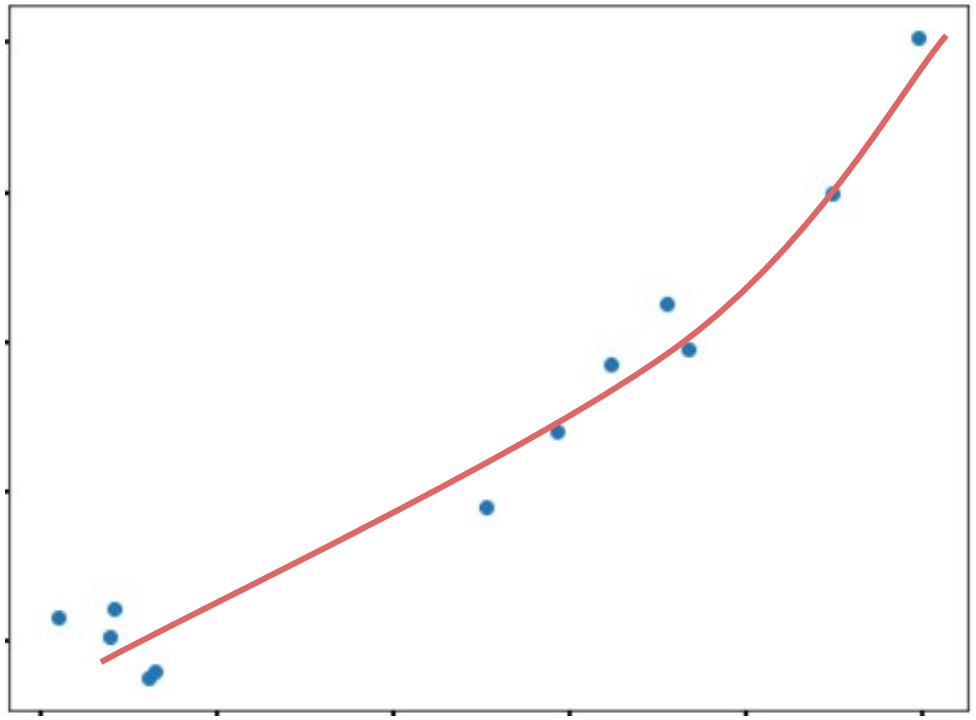
假設...

train data error = 15.2

test data error = 18.7

於是就讓function越來越複雜

...



Supervised Learning - Regression

Reconstruct Model

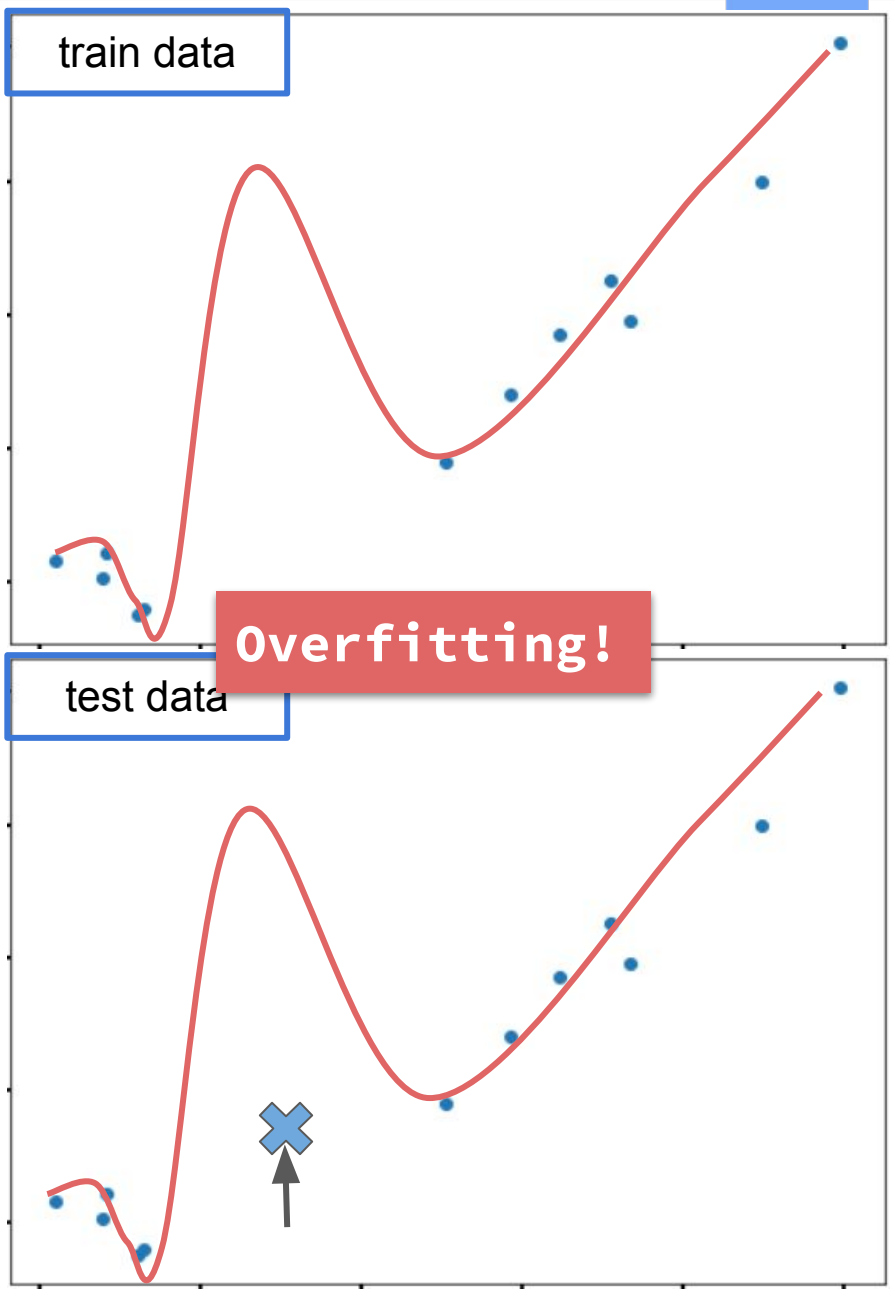
3次式 $y = w_3x^3 + w_2x^2 + w_1x + b$

4次式 $y = w_4x^4 + w_3x^3 + w_2x^2 + w_1x + b$

5次式 $y = w_5x^5 + w_4x^4 + w_3x^3 + w_2x^2 + w_1x + b$

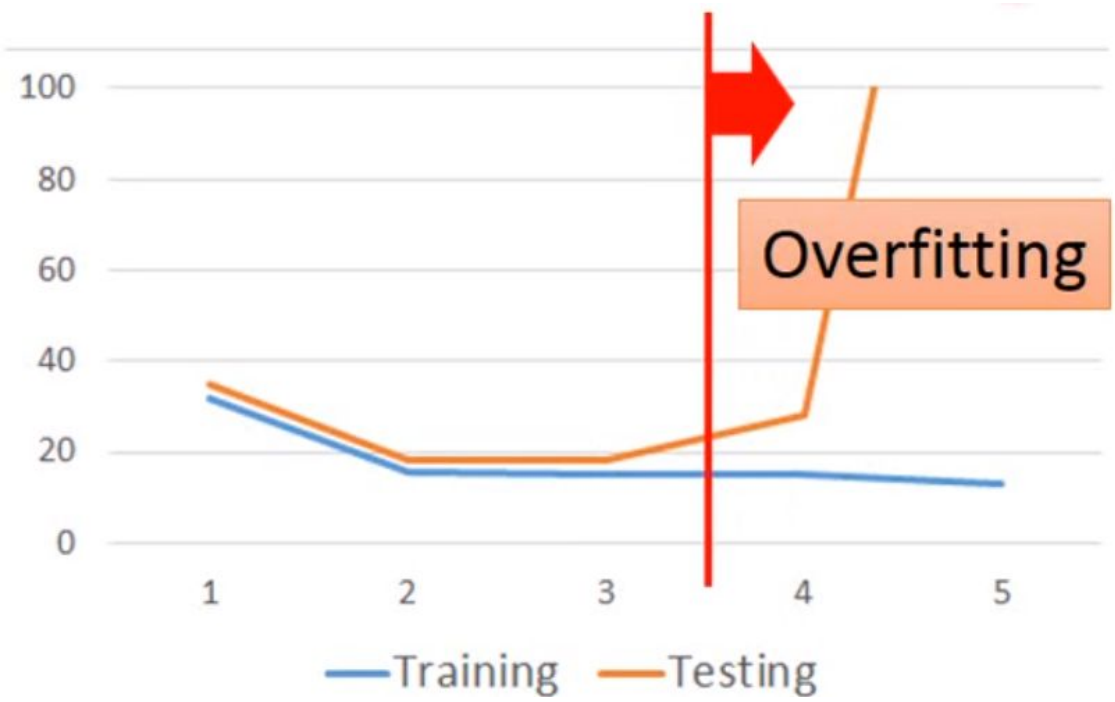
通常事情不會這們順利，右圖下方為test data，資料分布並沒有差很多

⇒ test loss >> train loss



Supervised Learning - Regression

Model Selection



越複雜的model也許可以在train data上越有好結果，但是不會代表在test data上也是

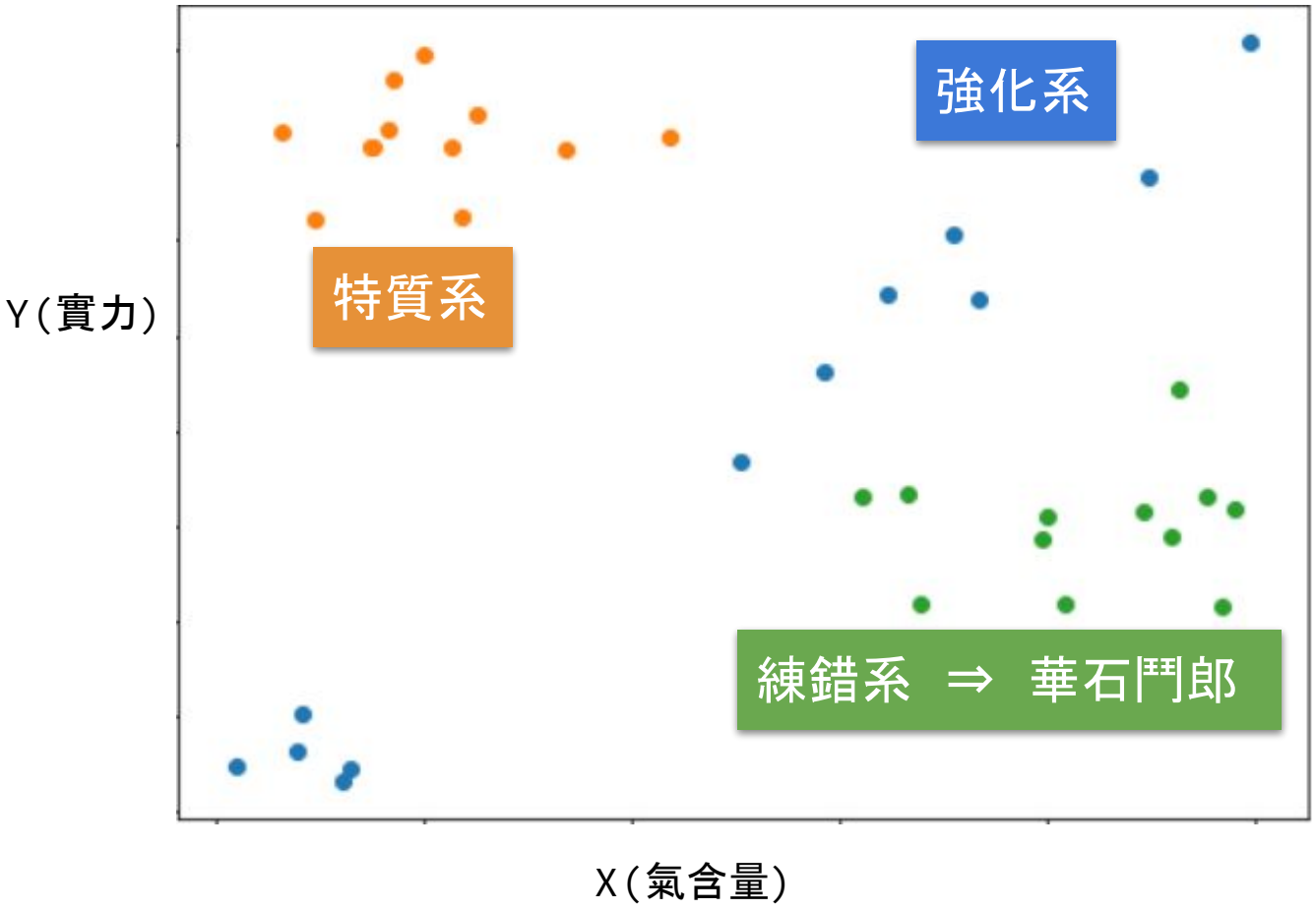
以左圖來說，我們會選擇3次式function的Model!

Supervised Learning - Regression

Maybe Another Hidden Factors

⇒ **加入系別判定(強化系，特質系...)**

加入更多資料!



Supervised Learning - Regression

Redesign The model

$$y = \sum w_i x_i + b$$

Linear Model?

x_1= 氣量(念基本功)，x_2= 系別

if x_2=	強化系	$y = b_1 + w_1 x_1$
if x_2=	變化系	$y = b_2 + w_2 x_1$
if x_2=	放出系	$y = b_3 + w_3 x_1$
if x_2=	具現化系	$y = b_4 + w_4 x_1$
if x_2=	操作系	$y = b_5 + w_5 x_1$
if x_2=	特質系	$y = b_6 + w_6 x_1$

要怎麼把if else寫成function?

Supervised Learning - Regression

Redesign The model

$$y = \sum w_i x_i + b$$

Linear Model?

y =

$\delta(x_2 = \text{操作系})(w_5 x_5 + b_5)$ +

$\delta \Rightarrow$ indicator

$=1$ if x_2 = 操作系

$=0$ otherwise

if x_2 = 操作系:

$\Rightarrow w_5 x_5 + b_5$

根據不同的條件有不同的function

Redesign The model

$$y = b_1 + w_1 x_1 + (w_2 x_2) + (w_3 x_3) + (w_4 x_4) + (w_5 x_5) + (w_6 x_6) + (w_7 x_7)$$

Alternative way to implement

	x_2	x_3	x_4	x_5	x_6	x_7
強化系	1	0	0	0	0	0
變化系	0	1	0	0	0	0
放出系	0	0	1	0	0	0
具現化系	0	0	0	1	0	0
操作系	0	0	0	0	1	0
特質系	0	0	0	0	0	1

if 操作系:

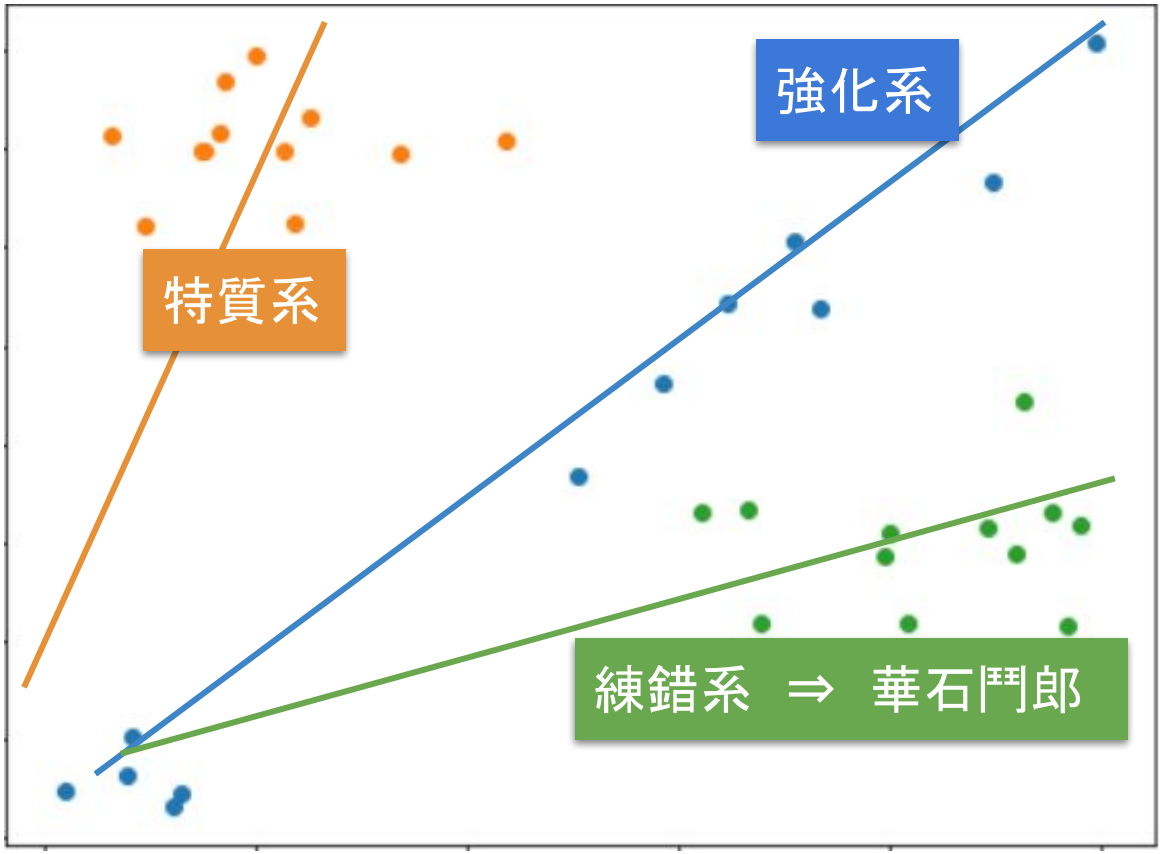
$\Rightarrow w_6 x_6 + w_1 x_1 + b_1$

One Hot Encoding

根據不同的條件有不同的function

Redesign The model

於是，看起來像這樣...



- **很多function在預測資料**
- 也可以加入2次式以上的作法

Supervised Learning - Regression(Summary)

Recall

- Functions set: $y = \sum w_i x_i + b$
- Loss function: $\sum (\hat{y}_i - (wx_i + b))^2$

- Optimizer: Gradient Descent
- 嘗試多次項，增加Model Performance
- Overfitting and Model Selection
- Categorical Variable ⇒ One Hot Encoding

Supervised Learning

Classification:
Logistic Regression

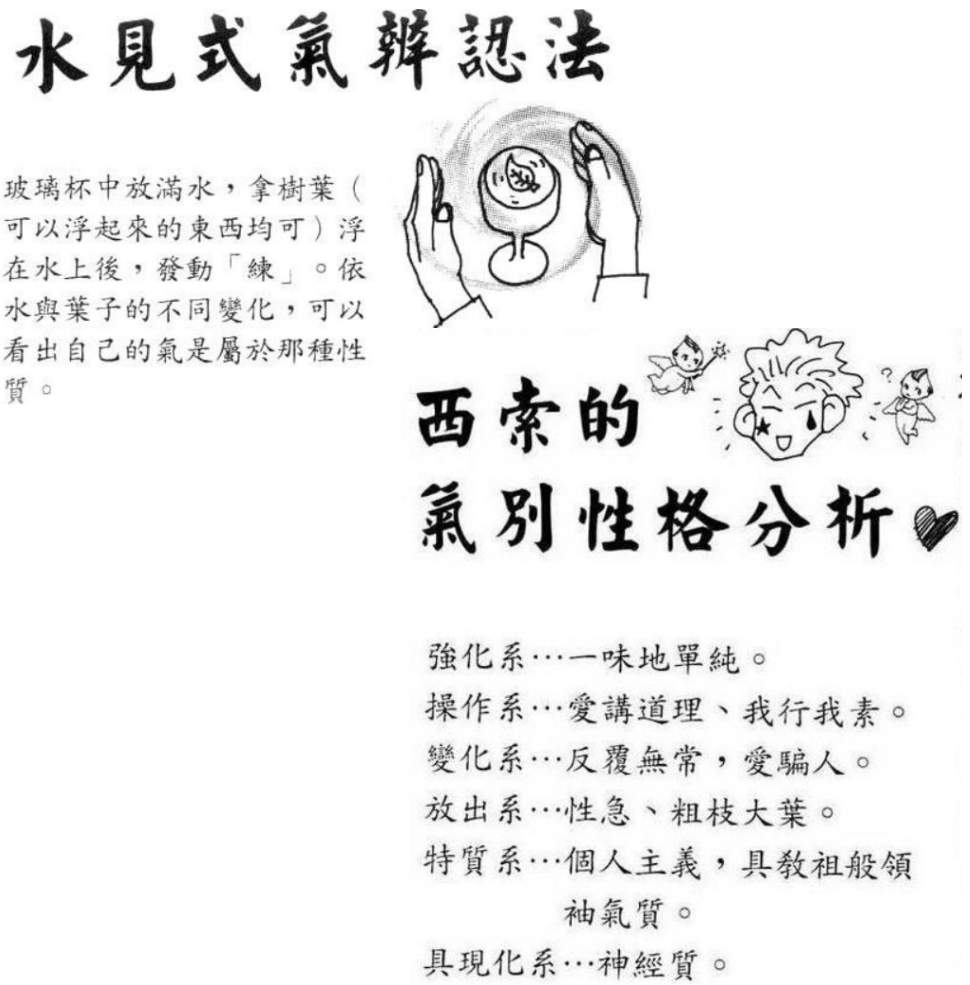
Supervised Learning - Classification

Logistic Regression

Binay Classification: 是否是強化系?

x_1	氣量	continuous
x_2 (水量變多)	水見式	category
⋮		
x_7 (水變甜)		
x_8 (一位的單純)	西索個性分析	category
⋮		
x_{13} (愛騙人)		
y	是否為強化系	label

Input: x1 ~ x13(用13個屬性描述)
Output: 是否是強化系?

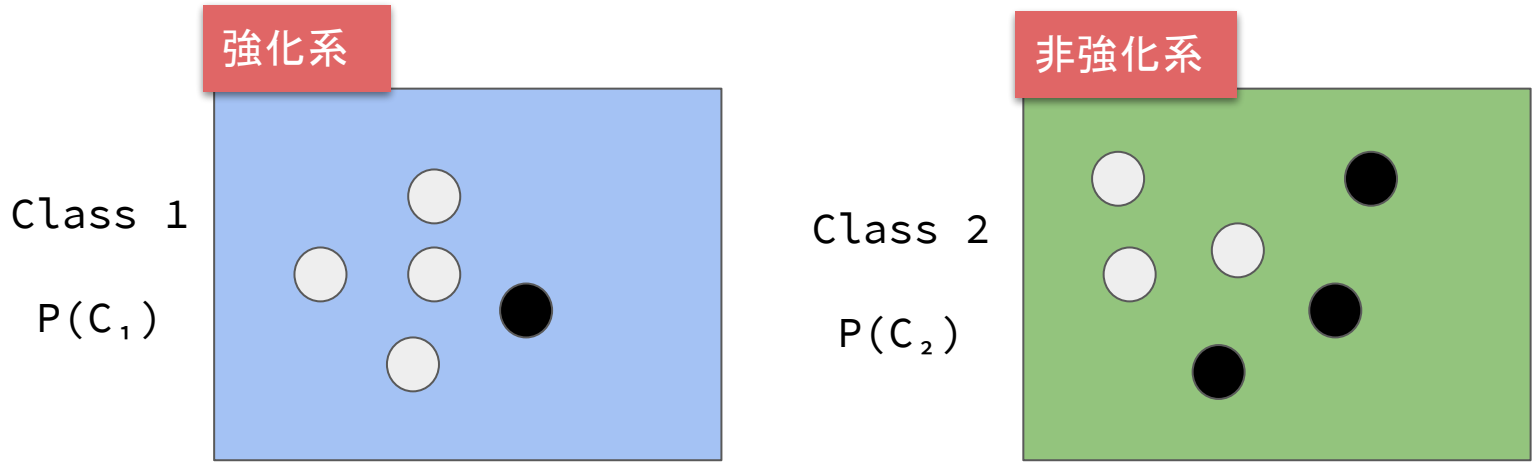


Supervised Learning - Classification

Logistic Regression

強項: Binay Classification!

Probabilistic Generative Model



$P(C_1) = 5 / 11$
$P(C_2) = 6 / 11$
$P(black|C_1) = 1 / 5$
$P(black|C_2) = 3 / 6$

可是我們真正要算的是類似這樣
⇒ $P(C_1|black)$ or $P(C_1|white)$

x: black or white

$$P(C_1|x) = \frac{P(x|C_1)P(C_1)}{P(x|C_1)P(C_1) + P(x|C_2)P(C_2)}$$

If $P(C_1|x) > 0.5$, output: class 1

Otherwise, output: class 2

Logistic Regression

Probabilistic Generative Model

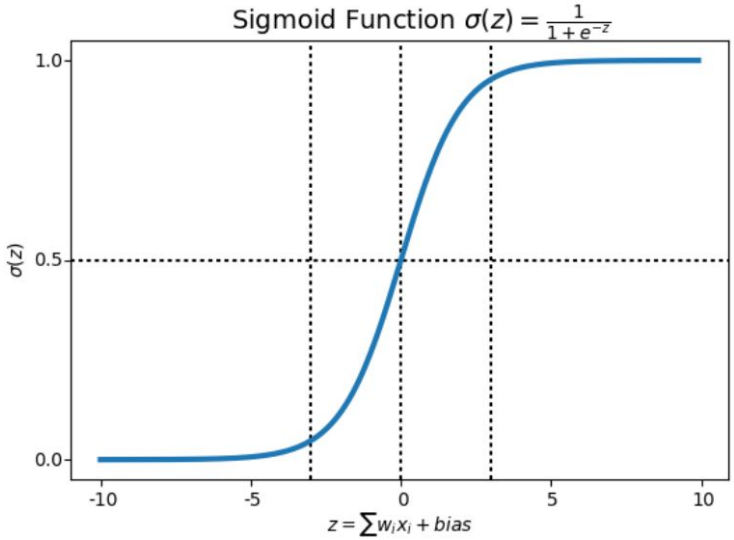
$$P(C_1|x) = \frac{P(x|C_1)P(C_1)}{P(x|C_1)P(C_1) + P(x|C_2)P(C_2)}$$

Activation function

$$= \frac{1}{1 + \frac{P(x|C_2)P(C_2)}{P(x|C_1)P(C_1)}} = \frac{1}{1 + exp(-z)}$$

$$z = ln \frac{P(x|C_1)P(C_1)}{P(x|C_2)P(C_2)}$$

log取負號等價於取倒數



Logistic Regression

Probabilistic Generative Model

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

$$P(C_1|x) = \sigma(w \cdot x + b)$$

$$z = w \cdot x + b$$

以wx + b估算z值，跟但能這麼做，是有經過證明的！

Step 1: Formulation

$$f_{w,b}(x) = \sigma\left(\sum_i w_i x_i + b\right)$$

Logistic Regression

Step2: Evaluate ⇒ Find Loss Function!

Training Data

x^1	x^2	x^3	……	x^N
C_1	C_1	C_2		C_1

假設每一筆資料屬於C_1的機率是base on $f_{w,b}(x) = P_{w,b}(C_1|x)$

Maximum Likelihood Estimation:
假設資料follow某種機率分布，必定有PDF or PMF可以計算機率，觀察N筆資料，找到參數(w, b)，讓這N筆資料發生的可能性最大！

➡ $L(w,b) = f_{w,b}(x^1) f_{w,b}(x^2) \left(1 - f_{w,b}(x^3)\right) \cdots f_{w,b}(x^N)$

➡ $w^*, b^* = arg \max_{w,b} L(w,b)$

假設每筆資料的機率都是獨立的

Logistic Regression

Step2: Evaluate ⇒ Find Loss Function!

x^1	x^2	x^3	……
$\hat{y}^1 = 1$	$\hat{y}^2 = 1$	$\hat{y}^3 = 0$	

$\hat{y}^n$: 1 for class 1, 0 for class 2

$$L(w,b) = f_{w,b}(x^1) f_{w,b}(x^2) \left(1 - f_{w,b}(x^3)\right) \cdots$$

$$w^*, b^* = arg \max_{w,b} L(w,b) = w^*, b^* = arg \min_{w,b} -lnL(w,b)$$

$$-lnL(w,b)$$

$$= -lnf_{w,b}(x^1) \rightarrow$$

$$-lnf_{w,b}(x^2) \rightarrow$$

$$-ln\left(1 - f_{w,b}(x^3)\right) \rightarrow$$

$$\vdots$$

Logistic Regression

Step2: Evaluate ⇒ Find Loss Function!

$$L(w,b) = f_{w,b}(x^1) f_{w,b}(x^2) \left(1 - f_{w,b}(x^3)\right) \cdots f_{w,b}(x^N)$$

$$-lnL(w,b) = lnf_{w,b}(x^1) + lnf_{w,b}(x^2) + ln\left(1 - f_{w,b}(x^3)\right) \cdots$$

$\hat{y}^n$: 1 for class 1, 0 for class 2

$$= \sum_n -\left[\hat{y}^n lnf_{w,b}(x^n) + (1-\hat{y}^n) ln\left(1 - f_{w,b}(x^n)\right)\right]$$

兩個Bernoulli分布的cross entropy

Label

Predict

Distribution p:

$p(x=1) = \hat{y}^n$

$p(x=0) = 1 - \hat{y}^n$

cross entropy

Distribution q:

$q(x=1) = f(x^n)$

$q(x=0) = 1 - f(x^n)$

Logistic Regression

Step3: Pick The Best Function

$$-\frac{lnL(w,b)}{\partial w_i} = \sum_n -\left[\hat{y}^n \frac{ln f_{w,b}(x^n)}{\partial w_i} + (1-\hat{y}^n) \frac{ln\left(1-f_{w,b}(x^n)\right)}{\partial w_i}\right]$$

$$\left(1-f_{w,b}(x^n)\right)x_i^n$$

$$\frac{\partial ln f_{w,b}(x)}{\partial w_i} = \frac{\partial ln f_{w,b}(x)}{\partial z}\frac{\partial z}{\partial w_i} \qquad \frac{\partial z}{\partial w_i} = x_i$$

$$\frac{\partial ln\sigma(z)}{\partial z} = \frac{1}{\sigma(z)}\frac{\partial \sigma(z)}{\partial z} = \frac{1}{\sigma(z)}\sigma(z)\left(1-\sigma(z)\right)$$

$$f_{w,b}(x) = \sigma(z)$$
$$= 1/1+exp(-z)$$

$$z = w \cdot x + b = \sum_i w_i x_i + b$$

Logistic Regression

Step3: Pick The Best Function

$$-\frac{\partial lnL(w,b)}{\partial w_i} = \sum_n -\left[\hat{y}^n \frac{\partial ln f_{w,b}(x^n)}{\partial w_i} + (1-\hat{y}^n)\frac{\partial ln\left(1-f_{w,b}(x^n)\right)}{\partial w_i}\right]$$

$$\frac{\partial ln f_{w,b}(x^n)}{\partial w_i} = \left(1-f_{w,b}(x^n)\right)x_i^n \qquad \frac{\partial ln\left(1-f_{w,b}(x^n)\right)}{\partial w_i} = -f_{w,b}(x^n)x_i^n$$

$$\frac{\partial ln\left(1-f_{w,b}(x)\right)}{\partial w_i} = \frac{\partial ln\left(1-f_{w,b}(x)\right)}{\partial z}\frac{\partial z}{\partial w_i} \qquad \frac{\partial z}{\partial w_i} = x_i$$

$$\frac{\partial ln(1-\sigma(z))}{\partial z} = \frac{1}{1-\sigma(z)}\frac{\partial \sigma(z)}{\partial z} = -\frac{1}{1-\sigma(z)}\sigma(z)(1-\sigma(z))$$

$$f_{w,b}(x) = \sigma(z)$$
$$= 1/1+exp(-z)$$

$$z = w\cdot x + b = \sum_i w_i x_i + b$$

Logistic Regression

Step3: Pick The Best Function

$$-\frac{\partial lnL(w,b)}{\partial w_i} = \sum_n -\left[\hat{y}^n \frac{\partial ln f_{w,b}(x^n)}{\partial w_i} + (1-\hat{y}^n)\frac{\partial ln\left(1-f_{w,b}(x^n)\right)}{\partial w_i}\right]$$

$\left(1-f_{w,b}(x^n)\right)x_i^n$ $\quad -f_{w,b}(x^n)x_i^n$

$$= \sum_n -\left[\hat{y}^n\left(1-f_{w,b}(x^n)\right)x_i^n - (1-\hat{y}^n)f_{w,b}(x^n)x_i^n\right]$$

$$= \sum_n -\left[\hat{y}^n - \hat{y}^n f_{w,b}(x^n) - f_{w,b}(x^n) + \hat{y}^n f_{w,b}(x^n)\right]x_i^n$$

$$= \sum_n -\left(\hat{y}^n - f_{w,b}(x^n)\right)x_i^n$$

Large difference large update

$$w_i \leftarrow w_i - \eta \sum_n -\left(\hat{y}^n - f_{w,b}(x^n)\right)x_i^n$$

Logistic Regression: Try Use Square Error!

Linear Regression我們沒有乘1/2,
加了1/2目的是要跟微分的平方抵銷！

$$L(f) = \frac{1}{2}\sum_n \left(f_{w,b}(x^n) - \hat{y}^n\right)^2$$

$$\frac{\partial \left(f_{w,b}(x)-\hat{y}\right)^2}{\partial w_i} = \left(f_{w,b}(x) - \hat{y}\right) \frac{\partial f_{w,b}(x)}{\partial z} \frac{\partial z}{\partial w_i}$$

$$= \left(f_{w,b}(x) - \hat{y}\right) f_{w,b}(x)\left(1 - f_{w,b}(x)\right) x_i$$

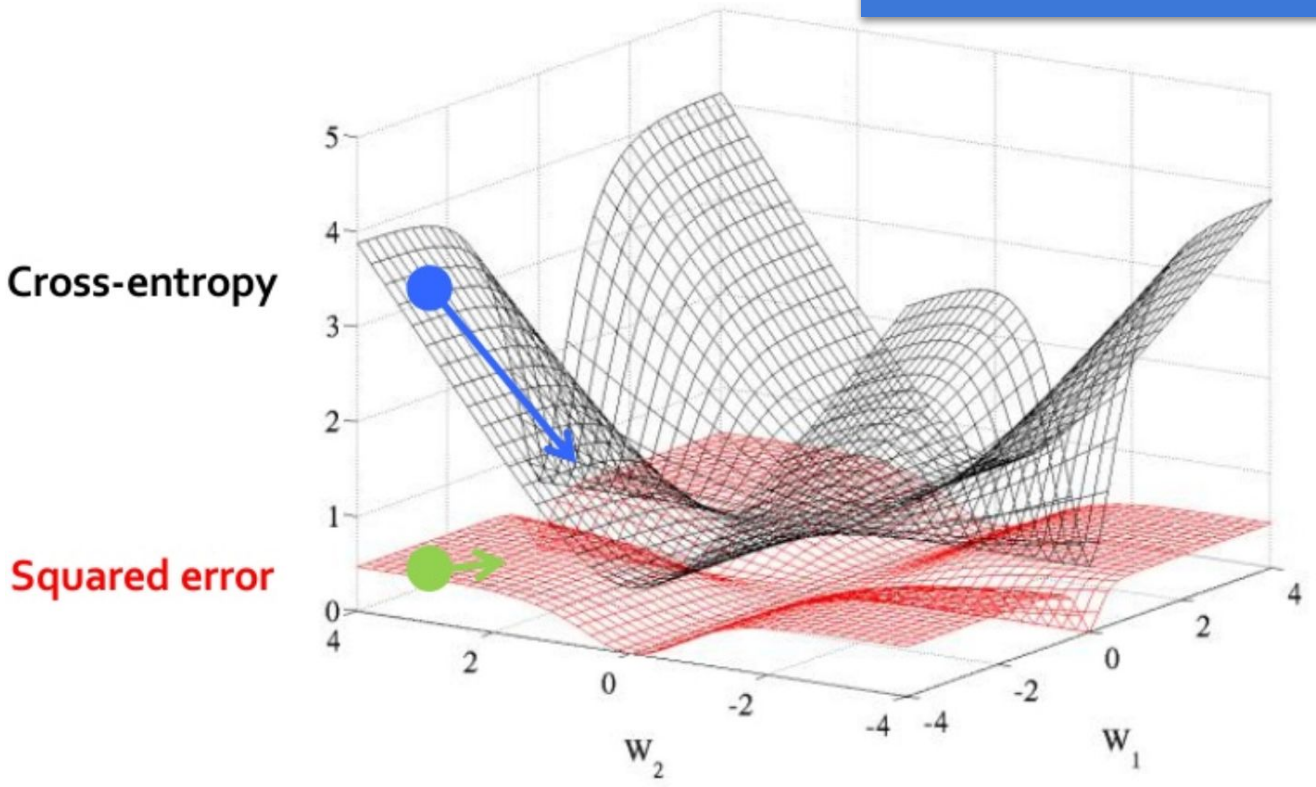
$\hat{y}^n = 1$ If $f_{w,b}(x^n) = 1$ (close to target) ➡ $\partial L/\partial w_i = 0$

If $f_{w,b}(x^n) = 0$ (far from target) ➡ $\partial L/\partial w_i = 0$

小提醒：$\hat{y}^n \in \{0, 1\}$ $f_{w,b}(x^n)$ *between* 0 *and* 1

Logistic Regression

Cross Entropy Good For Classification!



The error surface of logarithmic functions is steeper than that of quadratic functions.

Supervised Learning - Classification

What About Multi-class Classification?

判斷一個獵人屬於哪一種派系：
強化系、變化系、放出系、具現化系、操作系、特質系 （6種）

Data	Label	
$x_1 \ldots x_{13}$	y	
...	1	強化系
...	2	變化系
...	3	放出系
...	4	具現化系
...	5	操作系
...	6	特質系

這樣有點問題，**各類別間產生了距離的關係...**

dist(強化系, 變化系) = 2 - 1 = 1
dist(強化系, 放出系) = 3 - 1 = 2

解決方式？

Supervised Learning - Classification

What About Multi-Class Classification?

判斷屬於哪一種派系：
強化系、變化系、操作系、放出系、具現化系、特質系 (6種)

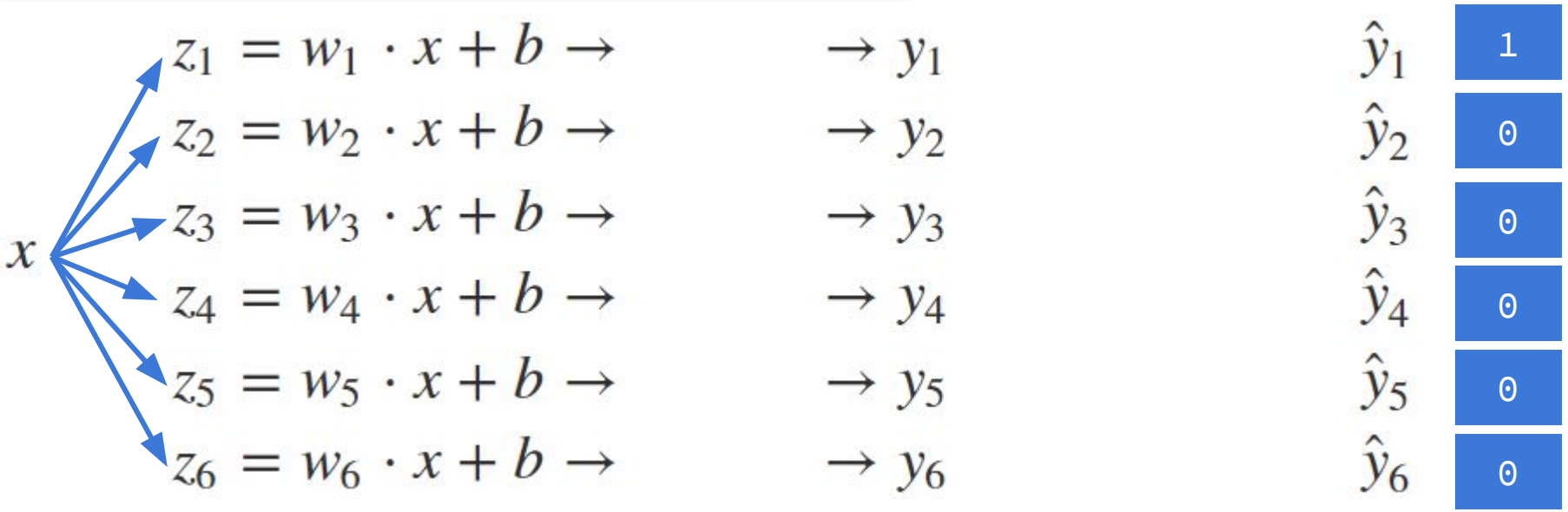
Data	Label(One Hot Encoding)						
$x_1 \ldots x_{13}$	y_1	y_2	y_3	y_4	y_5	y_6	
...	1	0	0	0	0	0	強化系
...	0	1	0	0	0	0	變化系
				⋮			⋮
...	0	0	0	0	0	1	特質系

Label的形式是Vector，怎麼處理?

Supervised Learning - Classification

What About Multi-Class Classification?

一千零一招：一個類別一個function對付他！



Probability:

- $1 > y_i > 0 \quad y_i = P(C_i \mid x)$
- $\sum_i y_i = 1$ → Sigmoid不滿足這個需求！

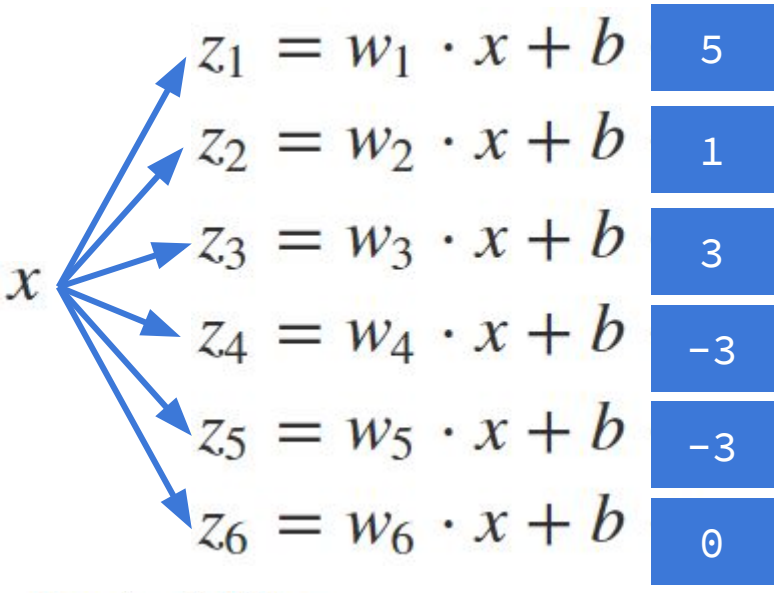
Supervised Learning - Classification

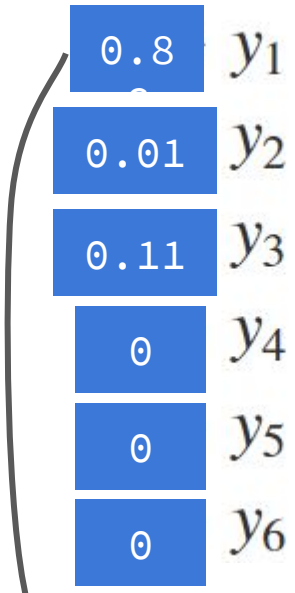
What About Multi-Class Classification?

使用Softmax

$$softmax(z) = \frac{e^{z_i}}{\sum_{k=1}^{n} e^{z_k}}$$

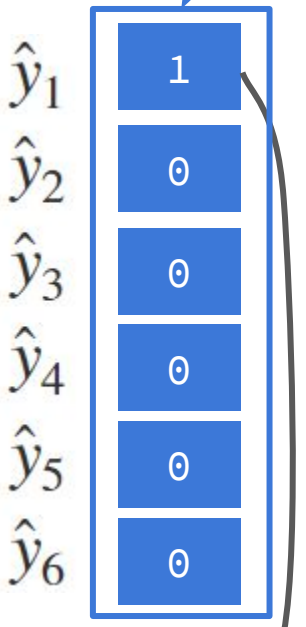
強化系





Cross Entropy

$$-\sum_{i=1} \hat{y}_i ln y_i$$



Probability:

- $1 > y_i > 0$ $\quad y_i = P(C_i \mid x)$
- $\sum_i y_i = 1$

計算argmax，最大的數字都在index 0 ⇒ 命中！

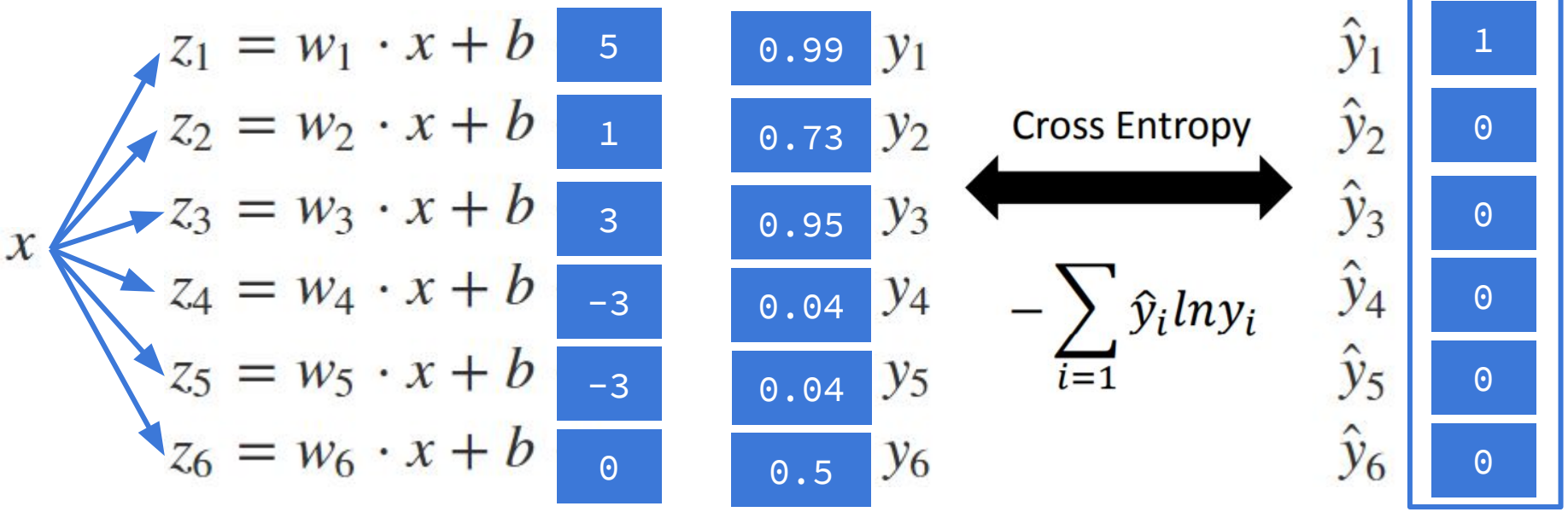
Supervised Learning - Classification

What About Multi-Class Classification?

Activation function
如果用sigmoid的話呢?

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

強化系



算argmax是沒有意義的!要考慮的是每個Class的分數

Supervised Learning - Classification

What About Multi-Class Classification?

Softmax v.s. Sigmoid

$$softmax(z) = \frac{e^{z_i}}{\sum_{k=1}^{n} e^{z_k}}$$

- ❏ 所有類別視為一個母體
- ❏ Single-label for one record

一個獵人只能屬於一個系別！

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

- ❏ 不考慮類別間的關係
- ❏ Multi-label for one record

一個獵人可以有多個系別，ex: 同時是強化系與特質系！

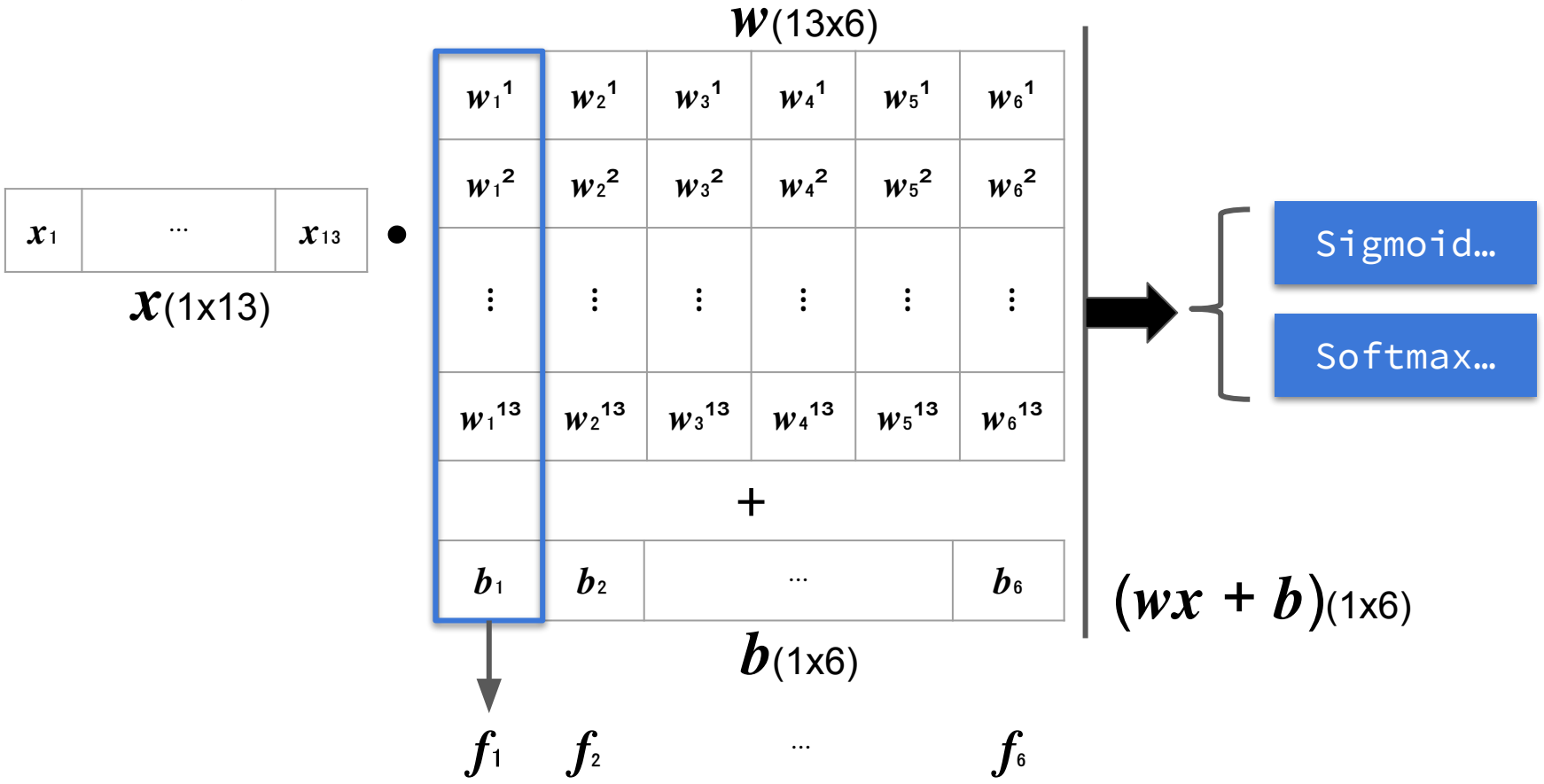
Computes sigmoid cross entropy given `logits`.

Measures the probability error in discrete classification tasks in which each class is independent and not mutually exclusive. For instance, one could perform multilabel classification where a picture can contain both an elephant and a dog at the same time.

Supervised Learning - Classification

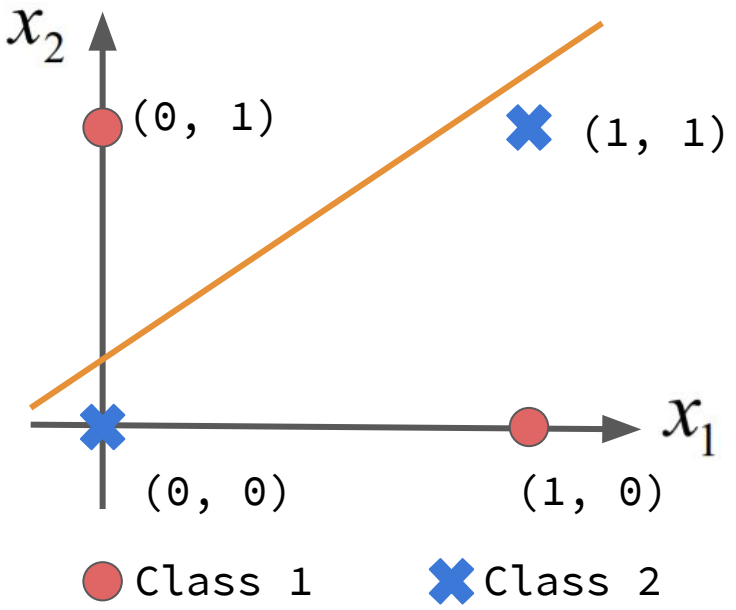
What About Multi-Class Classification?

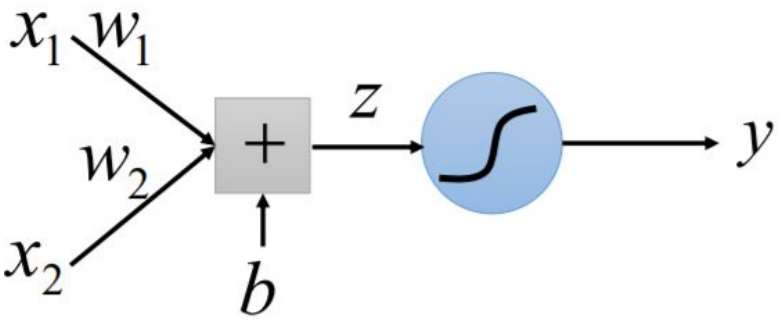
以向量運算表示~



Limitation of Logistic Regression

以下狀況似乎不能用一條線將他們分割...





$$\begin{cases} Class1 & y \geq 0.5 \\ Class2 & y < 0.5 \end{cases}$$

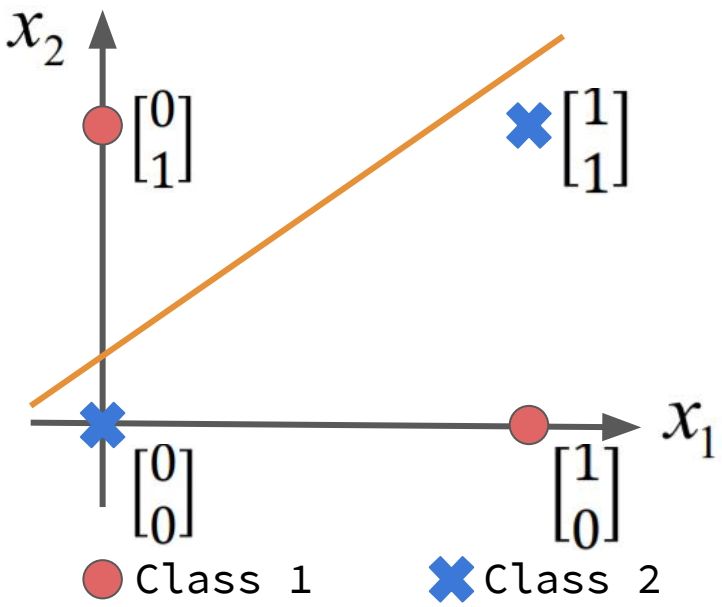
Limitation of Logistic Regression

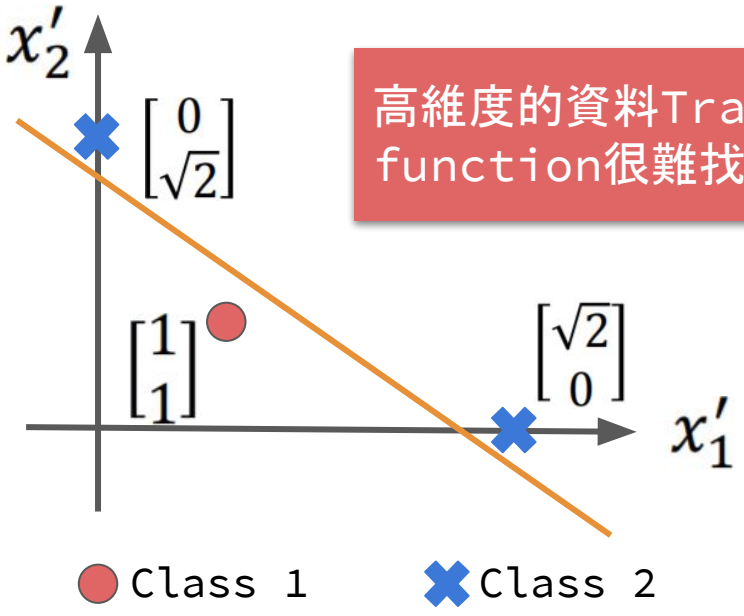
Feature Transformation!

$$trans(x) = \begin{bmatrix} x\text{到}(0,\ 0)\text{的距離} \\ x\text{到}(1,\ 1)\text{的距離} \end{bmatrix}$$

x_1': distance to $\begin{bmatrix} 0 \\ 0 \end{bmatrix}$

x_2': distance to $\begin{bmatrix} 1 \\ 1 \end{bmatrix}$

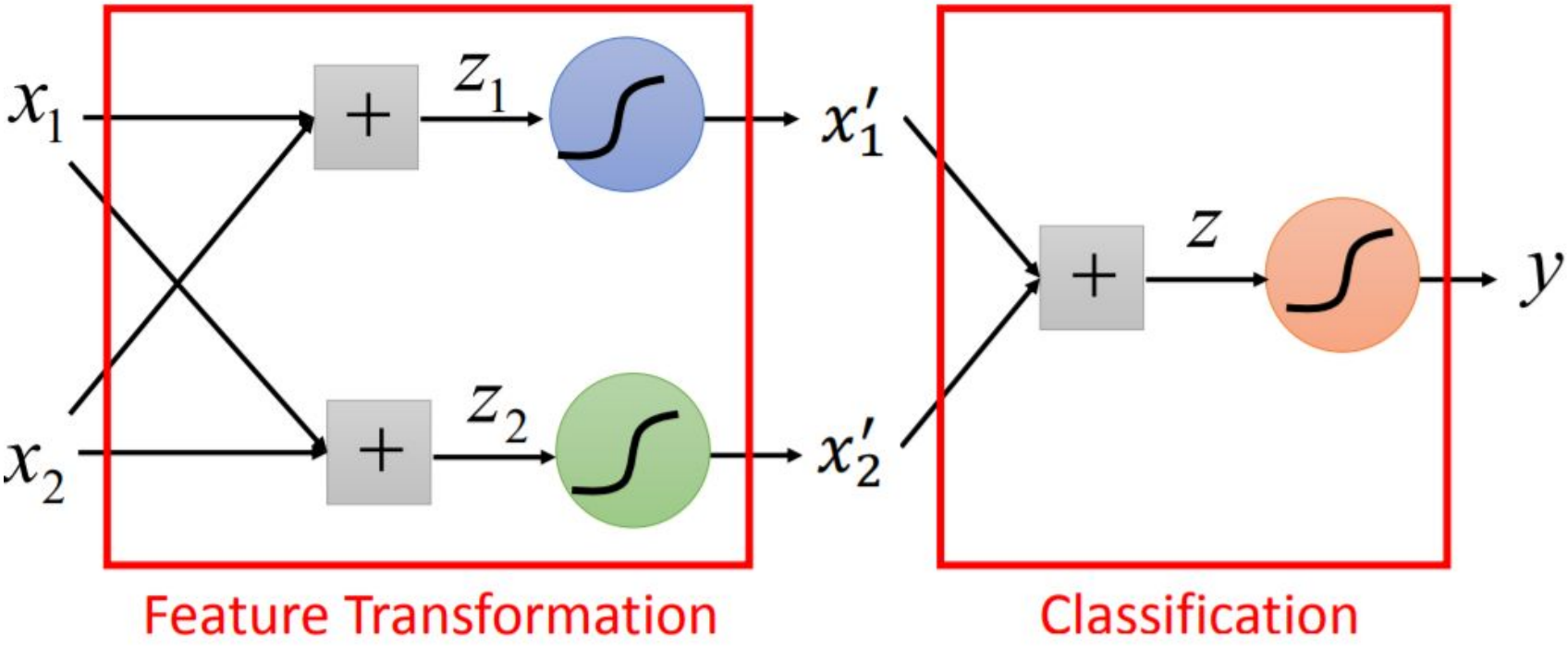




高維度的資料Transform function很難找...

Cascading Logistic Regression

也許Machine可以自己learn feature transformation…



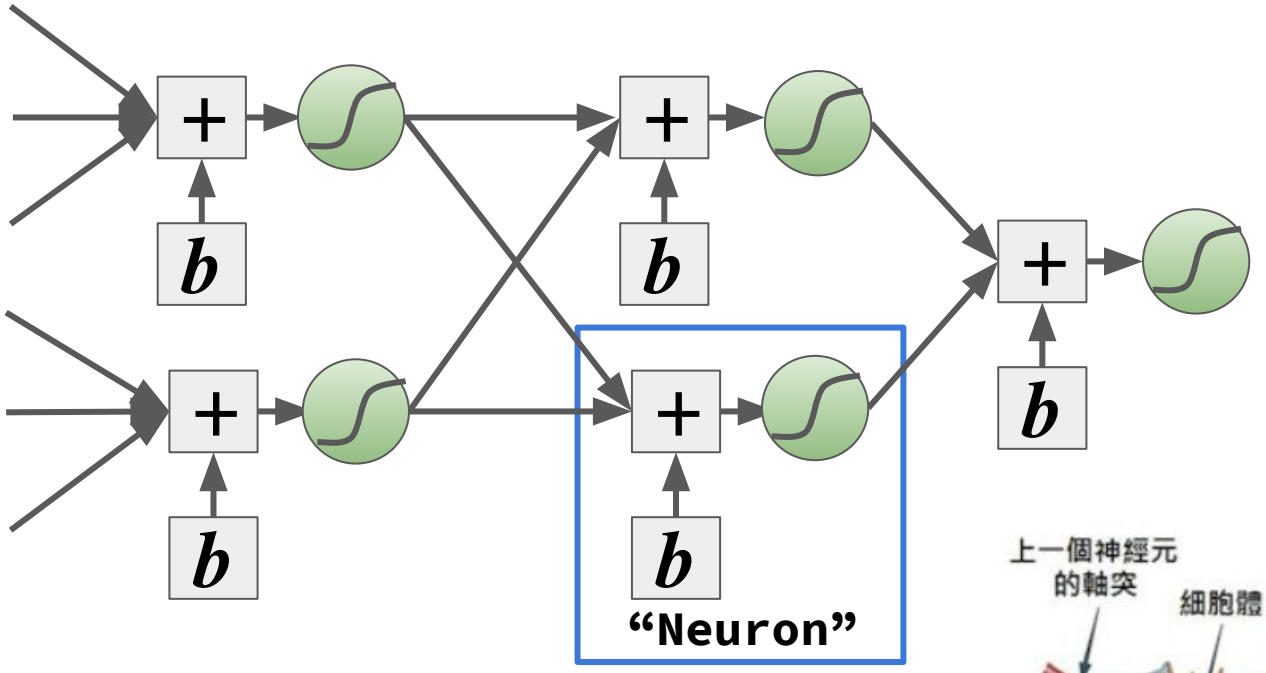
Supervised Learning - Classification

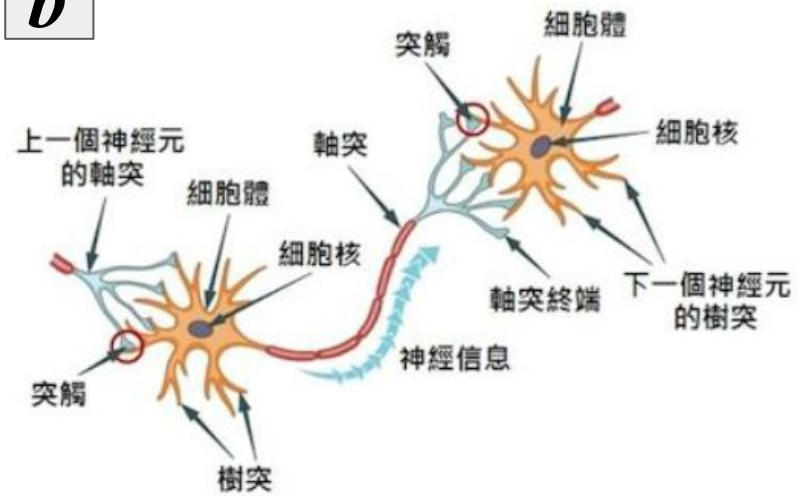
Cascading Logistic Regression

串聯 + 並聯

Deep Learning!

潮！





Supervised Learning - Classification(Summary)

Recall(1)

- Functions set: $\sigma(z) = \frac{1}{1 + exp(-z)} \quad z = w \cdot x + b$
- Posterior Probability Concept
- (Log)Maximum Likelihood Estimation
- Loss function: Cross Entropy
  - Why Cross Entropy Better than Squared Error(In Classification)
- Activation function:
  - Softmax for single-label classification
  - Sigmoid for multi-label classification
- Ensemble of Logistic Regression ⇒ Deep Learning!

Supervised Learning - Regression(Summary)

Recall(2)

	Logistic Regression	Linear Regression
Step 1	$f_{w,b}(x) = \sigma\left(\sum_i w_i x_i + b\right)$ Output: between 0 and 1	$f_{w,b}(x) = \sum_i w_i x_i + b$ Output: any value
Step 2	Label: 1 for Class 1, 0 for Class 2 $L(f) = -\sum_n \hat{y}^n \ lnf(x^n)$	Label: a real number $L(f) = \frac{1}{2}\sum_n (f(x^n) - \hat{y}^n)^2$
Step 3	$w_i \leftarrow w_i - \eta \sum_n -\left(\hat{y}^n - f_{w,b}(x^n)\right) x_i^n$	$w_i \leftarrow w_i - \eta \sum_n -\left(\hat{y}^n - f_{w,b}(x^n)\right) x_i^n$

Machine Learning 框架

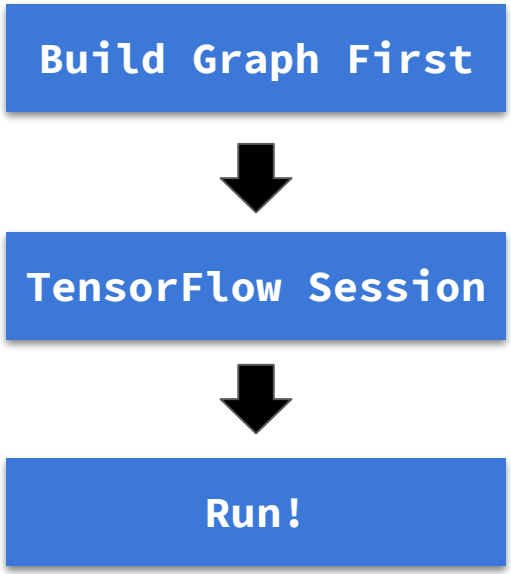
- Machine Learning Framework
- TensorFlow Fundamentals
- Deep Neural Network(DNN) Tips
- Recommedation Engine in Matrix Factorization
- Matrix Factorization with DNN

TensorFlow Fundamental

- 是Google開發的開源機器學習工具
- **透過使用Computational Graph，來進行數值演算。**
- 系統
  - Linux: Python2.7+、Python3+
  - MacOS: Python2.7+、Python3+
  - Windows: Python3.5+

Computational Graph

- A language describing a function
  - Node: varialbe(scalar, vetor, tensor…)
  - Edge: operation(四則運算, simple function)



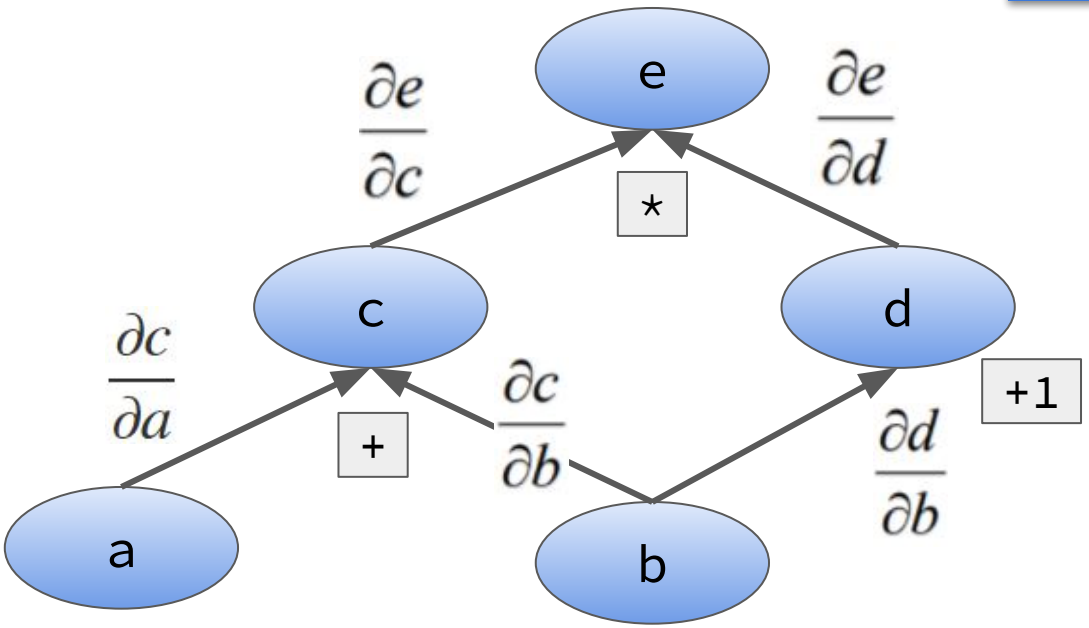
Computational Graph

- Example: e = (a+b)*(b+1)

```
c = a + b
d = b + 1
e = c * d
```

$$\frac{\partial e}{\partial b} = \frac{\partial e}{\partial c}\frac{\partial c}{\partial b} + \frac{\partial e}{\partial d}\frac{\partial d}{\partial b}$$

Chain Rule



TensorFlow Linear Regression Example

filename	tutorial_linear.ipynb
x	10000 of random generated value between 0 and 1
target function	y = 0.5x + 0.3
tensorflow function	y = **w**x + **b**
loss function	mean square error
learning rate	0.5
optimizer	tf.train.GradientDescentOptimizer

目標: 希望變數 w ⇒ 0.5, b ⇒ 0.3

小提醒: tensorflow並不知道data是由0.5x + 0.3產生的!

TensorFlow Linear Regression Example

Placeholder

```
g = tf.Graph()
with g.as_default():
    with tf.variable_scope("inputs"):
        # 接收 outerX data
        placeholderX = tf.placeholder(tf.float32, shape=[None], name="placeholder_x")
        # 接收 outerY data
        placeholderY = tf.placeholder(tf.float32, shape=[None], name="placeholder_y")
```

placeholderX	接收外部傳入資料 x (data)
placeholderY	接收外部傳入資料 y (label)

TensorFlow Linear Regression Example

TensorFlow formula graph

```
with tf.variable_scope("formula"):
    varW = tf.Variable(tf.random_uniform(shape=[1]), tf.float32, name="var_w")
    varB = tf.Variable(tf.random_uniform(shape=[1]), tf.float32, name="var_b")
    # 公式 0.5X + 0.3, 期望varW會慢慢逼近0.5, varB會慢慢逼近0.3
    varY = varW * placeholderX + varB
    # tensorboard weights logs
    tf.summary.histogram("varW", varW)
    tf.summary.histogram("varB", varB)
```

varW	initialize tensor variable by uniform distribution
varB	initialize tensor variable by uniform distribution
varY	**varW * placeholderX + varB**
tf.summary.histogram	紀錄參數變化

TensorFlow Linear Regression Example

TensorFlow training graph

```
# mean square error
loss = tf.losses.mean_squared_error(placeholderY, varY)
# tensorboard weights logs
tf.summary.scalar("loss", loss)
# 使用 GradientDescentOptimizer
optimizer = tf.train.GradientDescentOptimizer(learning_rate=learning_rate)
# 最小化 loss, 此時會 backpropagation 去調整 varW, varB的值
trainer = optimizer.minimize(loss)
merge = tf.summary.merge_all()
```

loss	mean squared error
trainer	training operation
tf.summary.scalar	紀錄loss變化
tf.summary.merge_all	合併所有tensorboard log變數

TensorFlow Linear Regression Example

TensorFlow session run

feed_dict參數將外部資料傳給placholder

```
with tf.Session(graph=g) as sess:
    w = tf.summary.FileWriter("./model/linear", sess.graph)
    init = tf.global_variables_initializer()
    sess.run(init)
    for i in range(n_epoch):
        _, w_, b_ = sess.run([trainer, varW, varB], feed_dict={placeholderX: outerX, placeholderY: outerY})
        print('step: {} varW: {}, varB: {}'.format(i, w_, b_) )
        if (i + 1) % 20 == 0:
            merge_ = sess.run(merge, feed_dict={placeholderX: outerX, placeholderY: outerY})
            w.add_summary(merge_, i)
```

init variable

session run取值，別忘了run trainer operation

Tensorboard log

w: file writer
w.add_summary: 在loop中控制log freq
logdir: ./model/linear

TensorFlow Linear Regression Example

如何啟動TensorBoard?

Local (cmd啟動web server, port:6006)

```
tensorboard --logdir="指定資料夾"
```

GCP Datalab (call datalab machine learning API)

```
from google.datalab.ml import TensorBoard

tb = TensorBoard()
# stop prev built tensorboard pid
for _, r in tb.list().iterrows(): tb.stop(r.pid)

tb.start(model_dir)
tb.list()
```

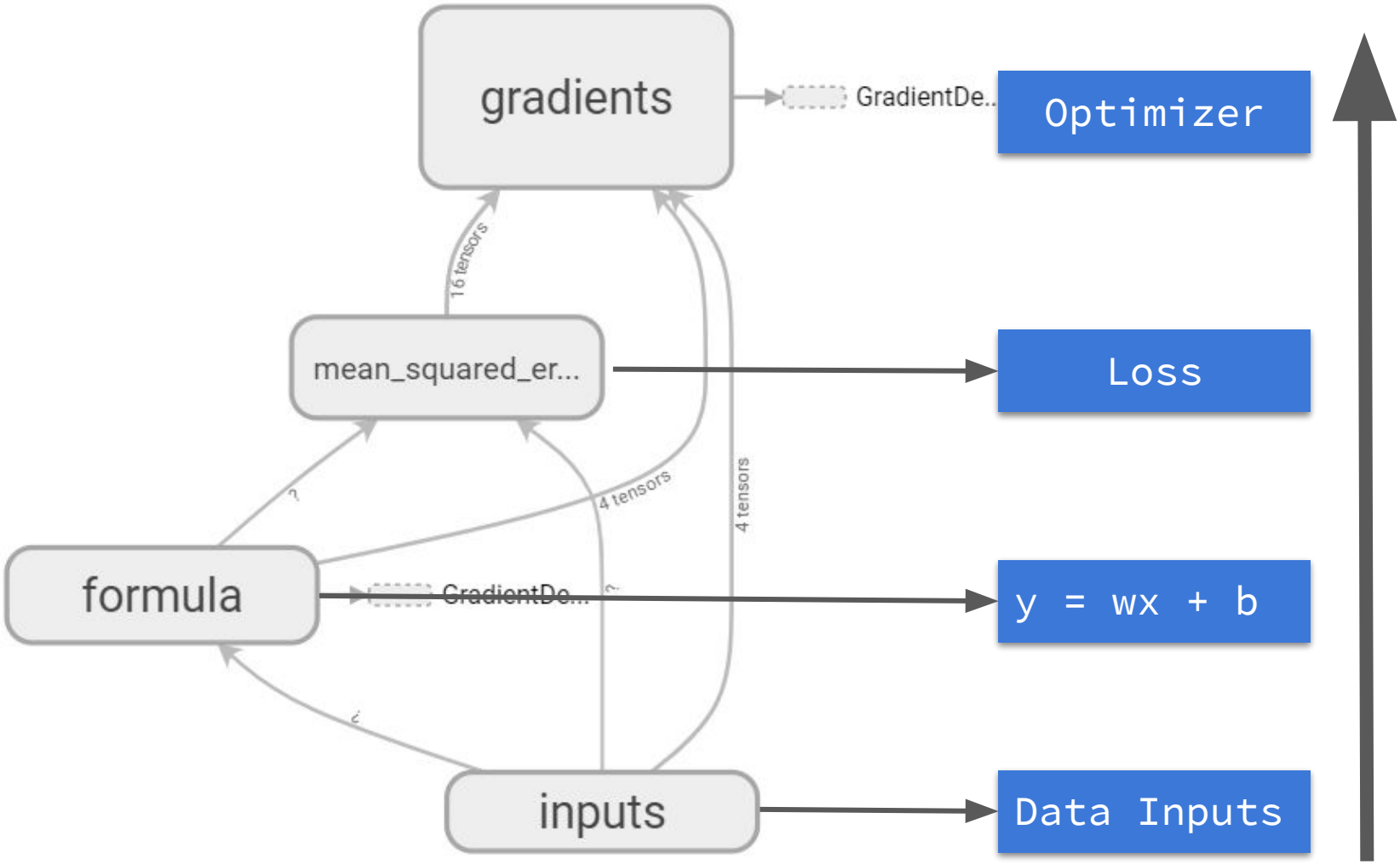
GCP Datalab無法開啟 port 6006

TensorBoard was started successfully with pid 9595. Click here to access it.

	logdir	pid	port
0	./model/linear	9595	50667

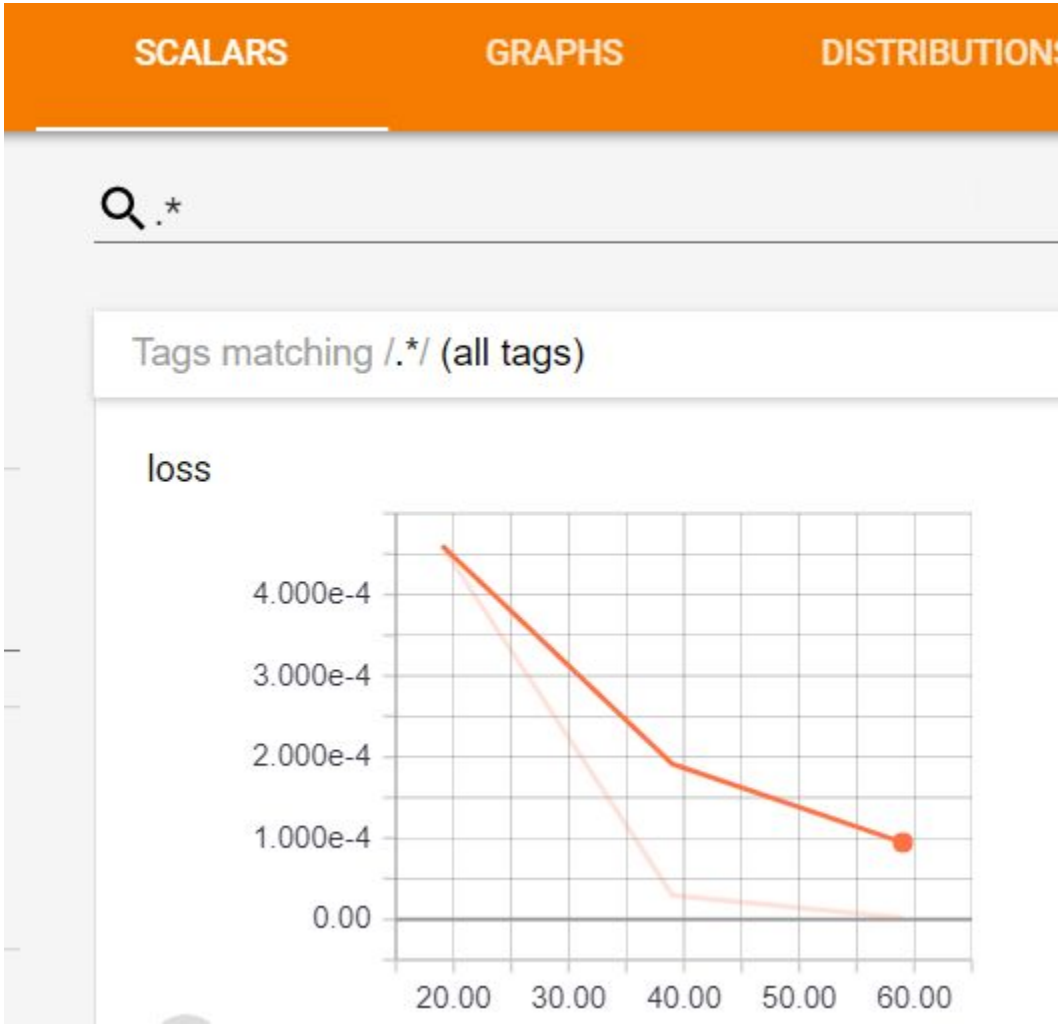
TensorFlow Linear Regression Example

TensorBoard for this lab



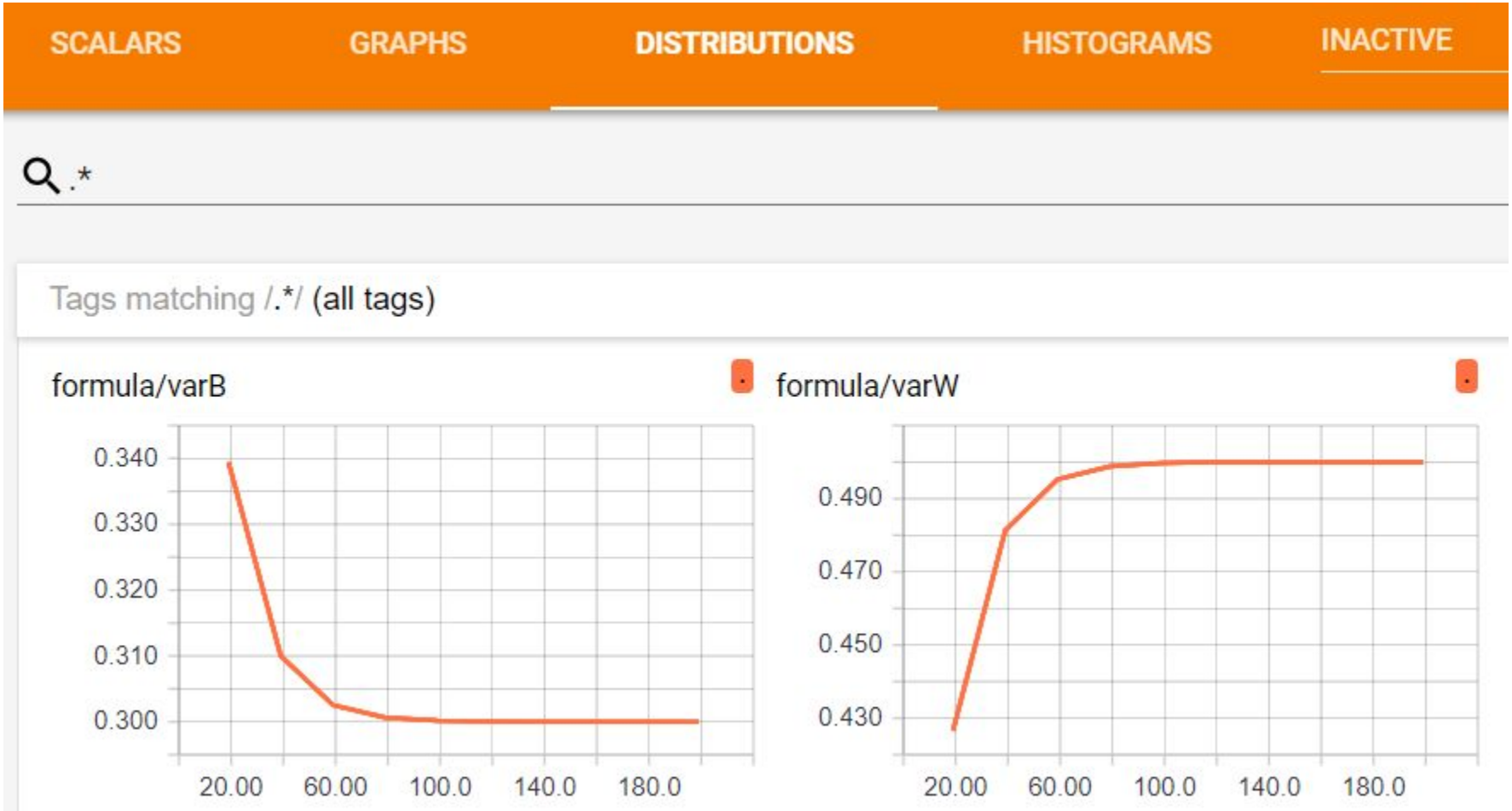
TensorFlow Linear Regression Example

TensorBoard: loss history



TensorFlow Linear Regression Example

TensorBoard: varW and varB history



TensorFlow Linear Regression Example

TensorFlow model save

Build saver object in Graph

```
# 儲存物件
self.saver = tf.train.Saver(tf.global_variables())
```

Save in Session runtime

```
if (i + 1) % 50 == 0:
    ''' Save model '''
    model.saver.save(sess, save_path=model_dir + "/linear_model", global_step=i)
```

每50 steps存檔一次，global_step參數是存檔index，不給的話就一會覆蓋前一次checkpoint

TensorFlow Linear Regression Example

TensorFlow load model

Load model pre-trained weights

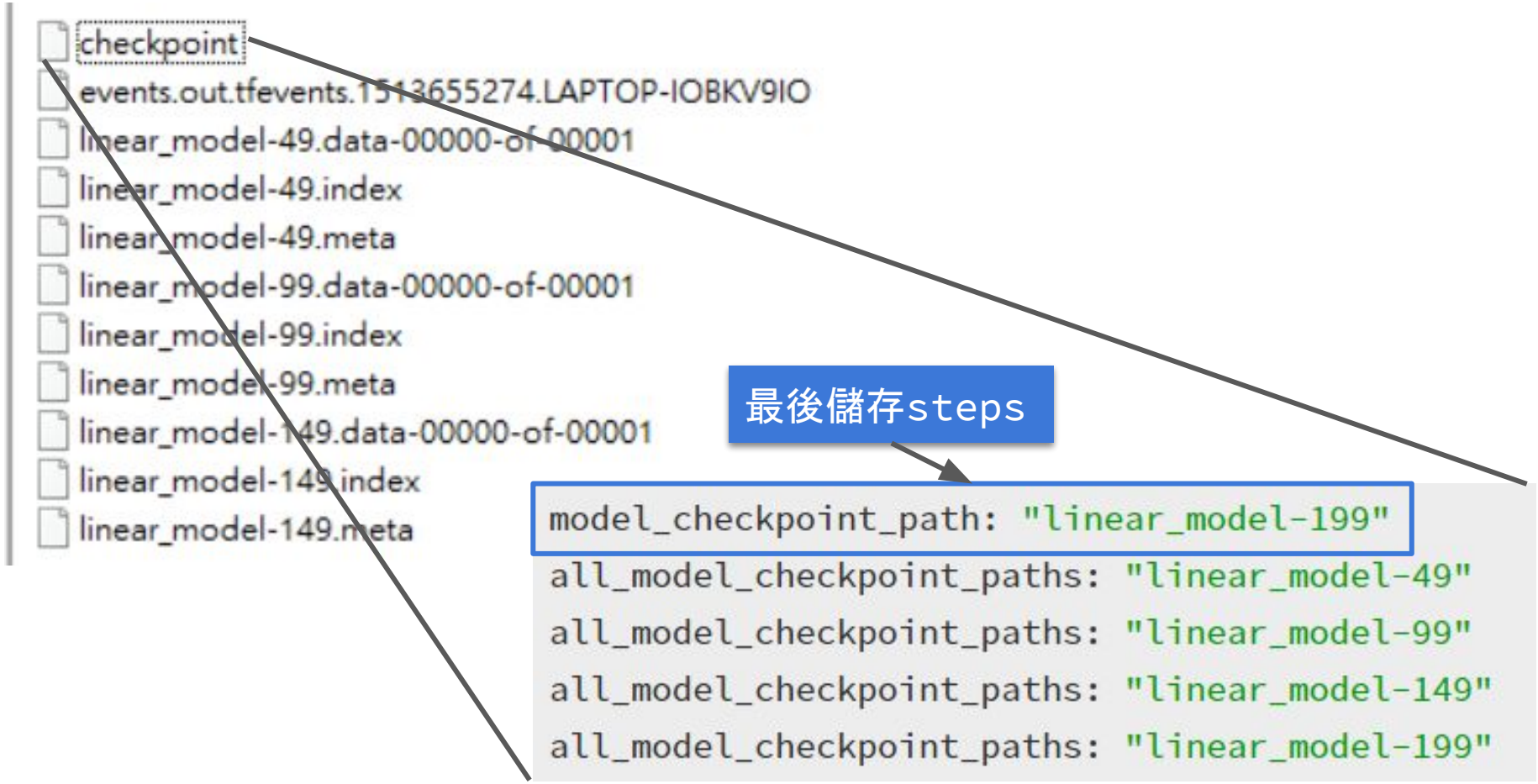
```
model = LinearModel()
with tf.Session(graph=model.graph) as sess:
    sess.run(tf.global_variables_initializer())
    ''' load pre-trained weight into model '''
    model.load_weight(sess, model_dir)
    ''' ################################## '''
```

Use tensorflow built-in fnction
tf.train.latest_checkpoint

```
def load_weight(self, sess, model_dir):
    """load latest saved model"""
    latestCkpt = tf.train.latest_checkpoint(model_dir)
    if latestCkpt:
        self.saver.restore(sess, latestCkpt)
        print("load weight success!")
    return latestCkpt
```

TensorFlow Linear Regression Example

TensorFlow load model



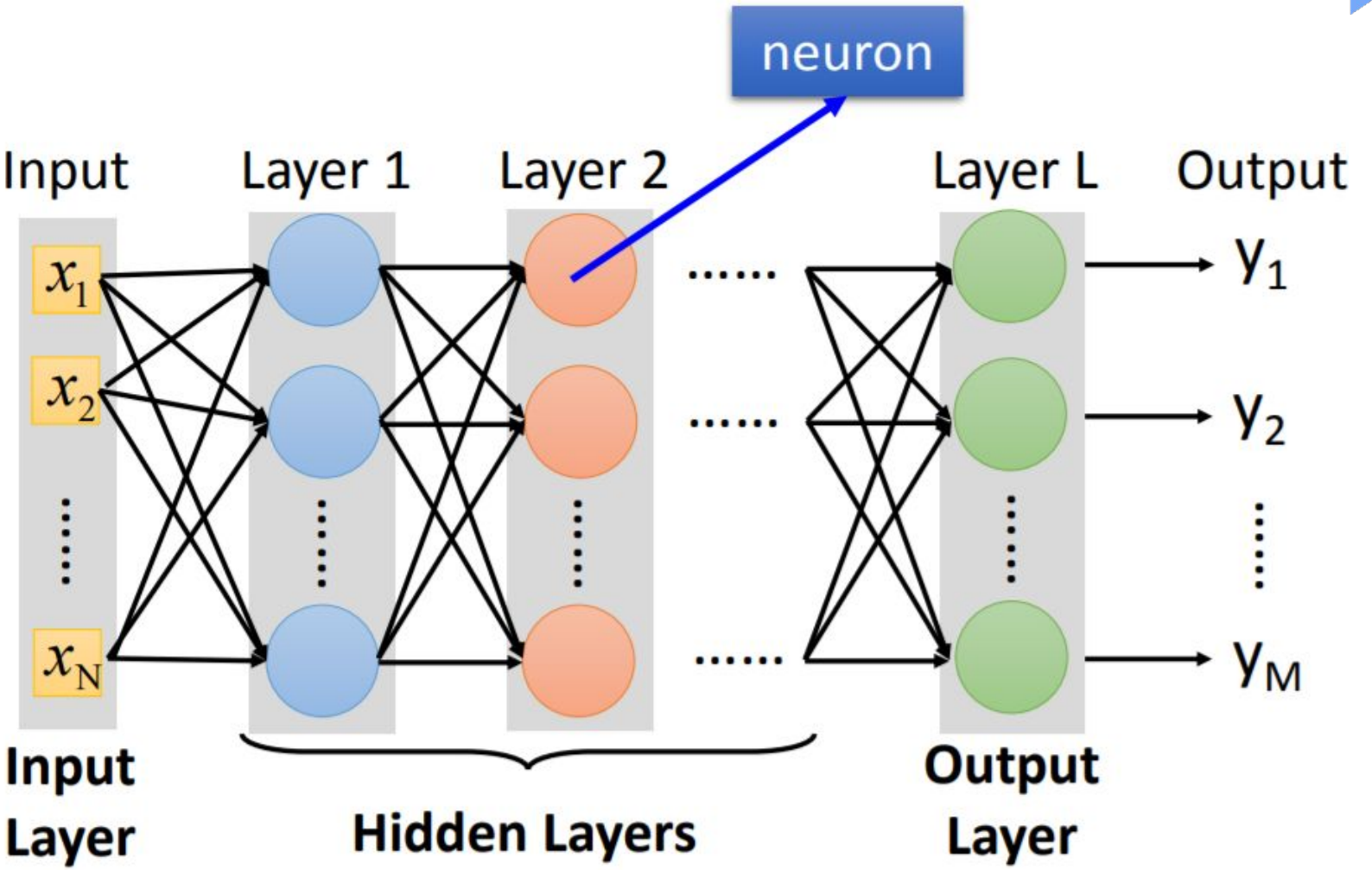
Lab: lab_tutorial_x_square.ipynb (10 - 15 minutes)

x	10000 of random generated value between 0 and 1
target function	$Y = 4X^2 + 8X + 7$
tensorflow function	y = **varA** * x **2 + **varB** * x + **varC**
loss function	mean square error
learning rate	0.5
optimizer	tf.train.GradientDescentOptimizer

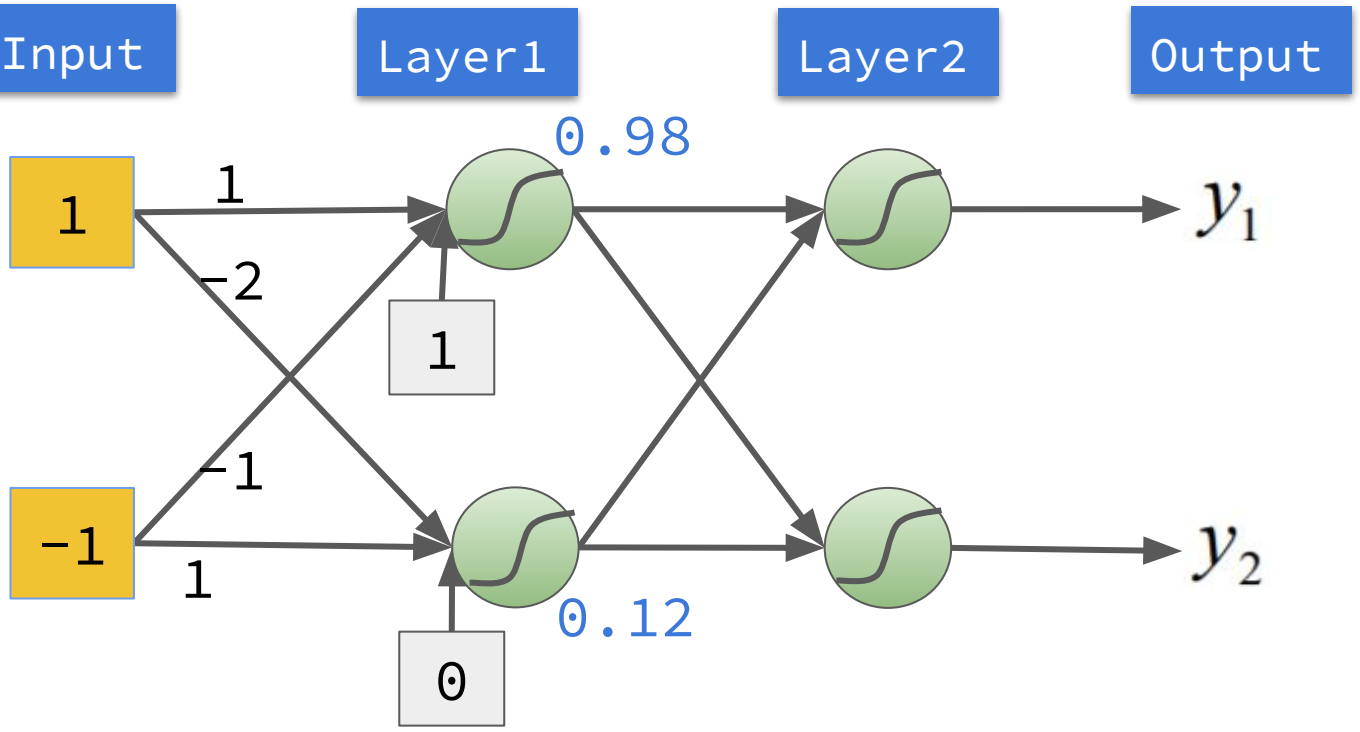
Machine Learning 框架

- Machine Learning Framework
- TensorFlow Fundamentals
- Deep Neural Network(DNN) Tips
- Recommedation Engine in Matrix Factorization
- Matrix Factorization with DNN

Deep Learning Structure(Fully Connected)



Matrix Operation



$$\sigma\left(\begin{bmatrix}1 & -1\end{bmatrix}\begin{bmatrix}1 & -1\\ -2 & 1\end{bmatrix}+\begin{bmatrix}1 & 0\end{bmatrix}\right)=\begin{bmatrix}0.98\\ 0.12\end{bmatrix}$$

data　weight　bias

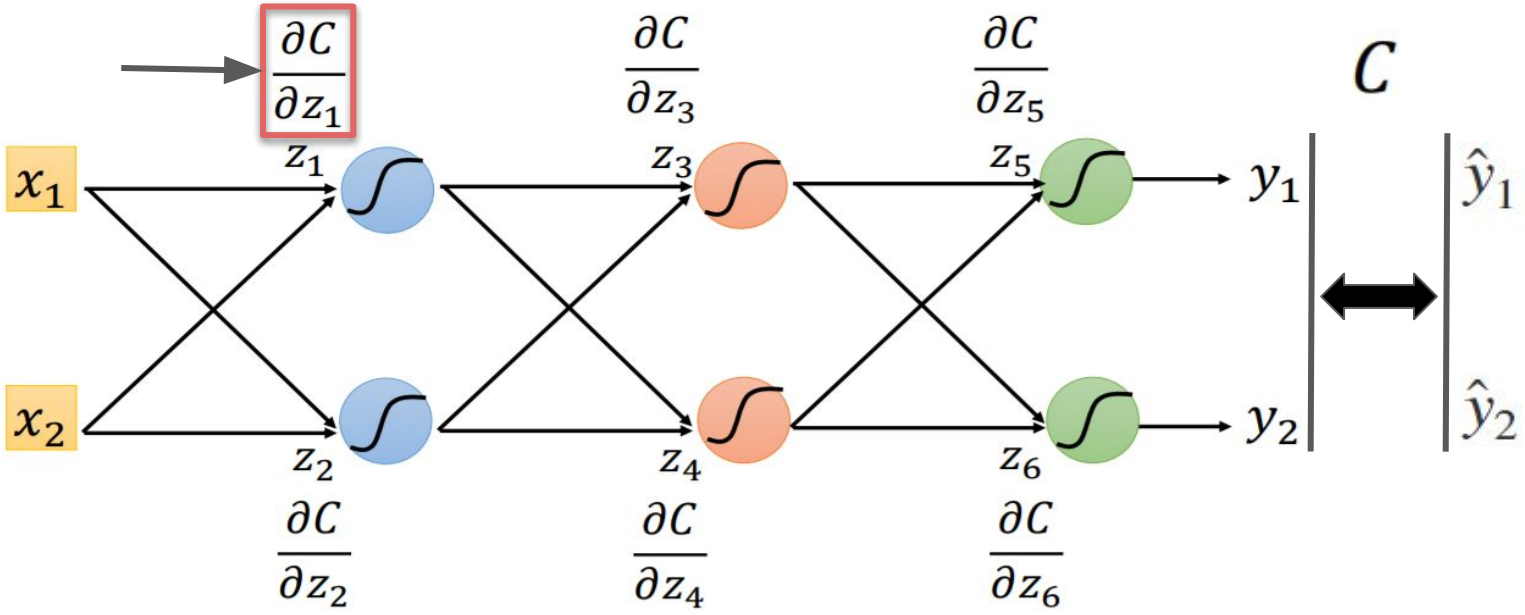
一層Layer的組成元素

σ, w, b

Backpropagation: Crazy Chain Rule

Compute $\partial C/\partial z$ for all activation function inputs z

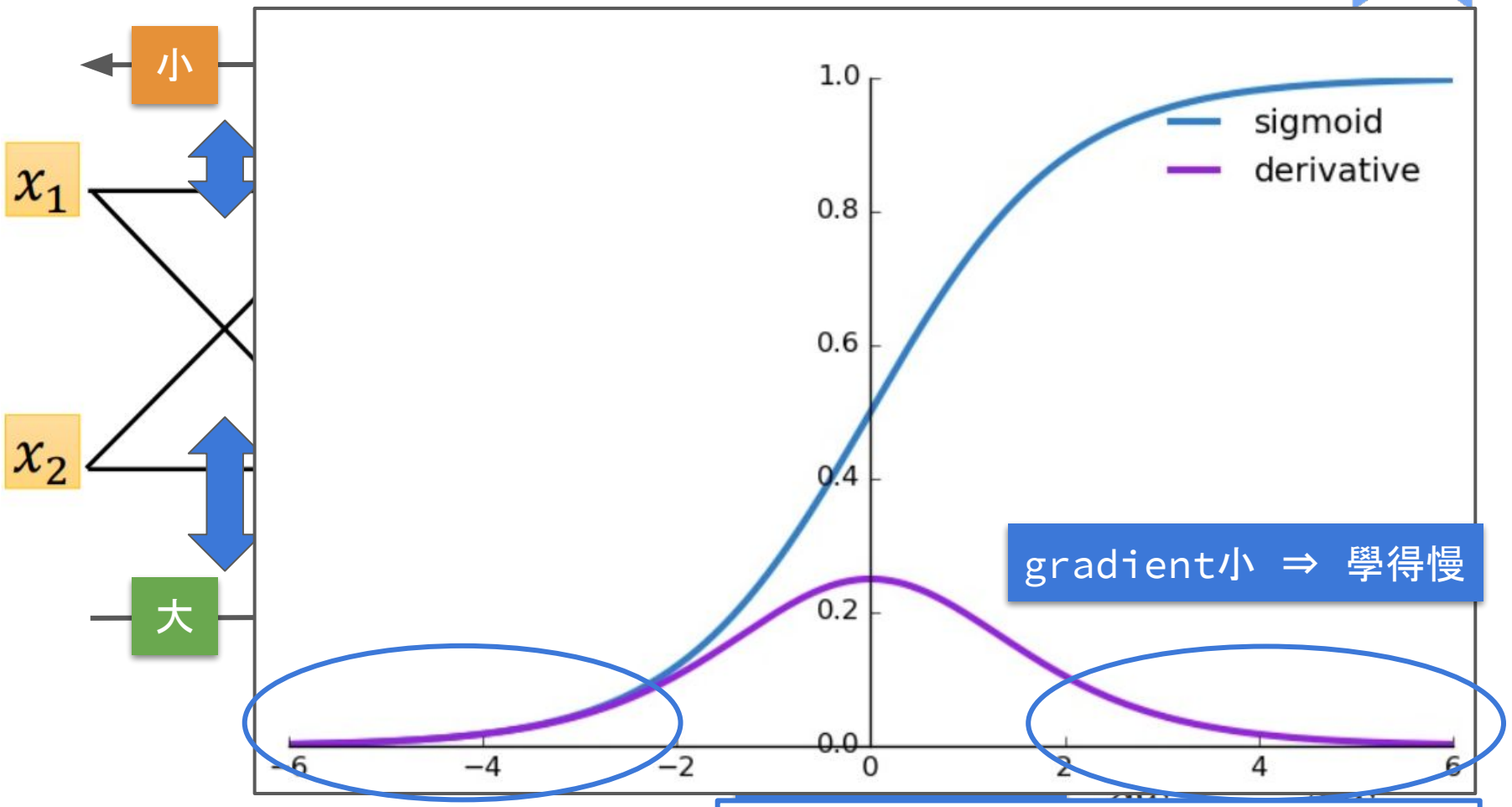
Compute $\partial C/\partial z$ from the output layer



$$\frac{\partial C}{\partial z_1} = \frac{\partial C}{\partial z_5}\frac{\partial z_5}{\partial z_3}\frac{\partial z_3}{\partial z_1} + \frac{\partial C}{\partial z_5}\frac{\partial z_5}{\partial z_4}\frac{\partial z_4}{\partial z_1} + \frac{\partial C}{\partial z_6}\frac{\partial z_6}{\partial z_3}\frac{\partial z_3}{\partial z_1} + \frac{\partial C}{\partial z_6}\frac{\partial z_6}{\partial z_4}\frac{\partial z_4}{\partial z_1}$$

$$\frac{\partial C}{\partial z_5} = \frac{\partial C}{\partial \sigma}\frac{\partial \sigma}{\partial z_5}\cdots$$

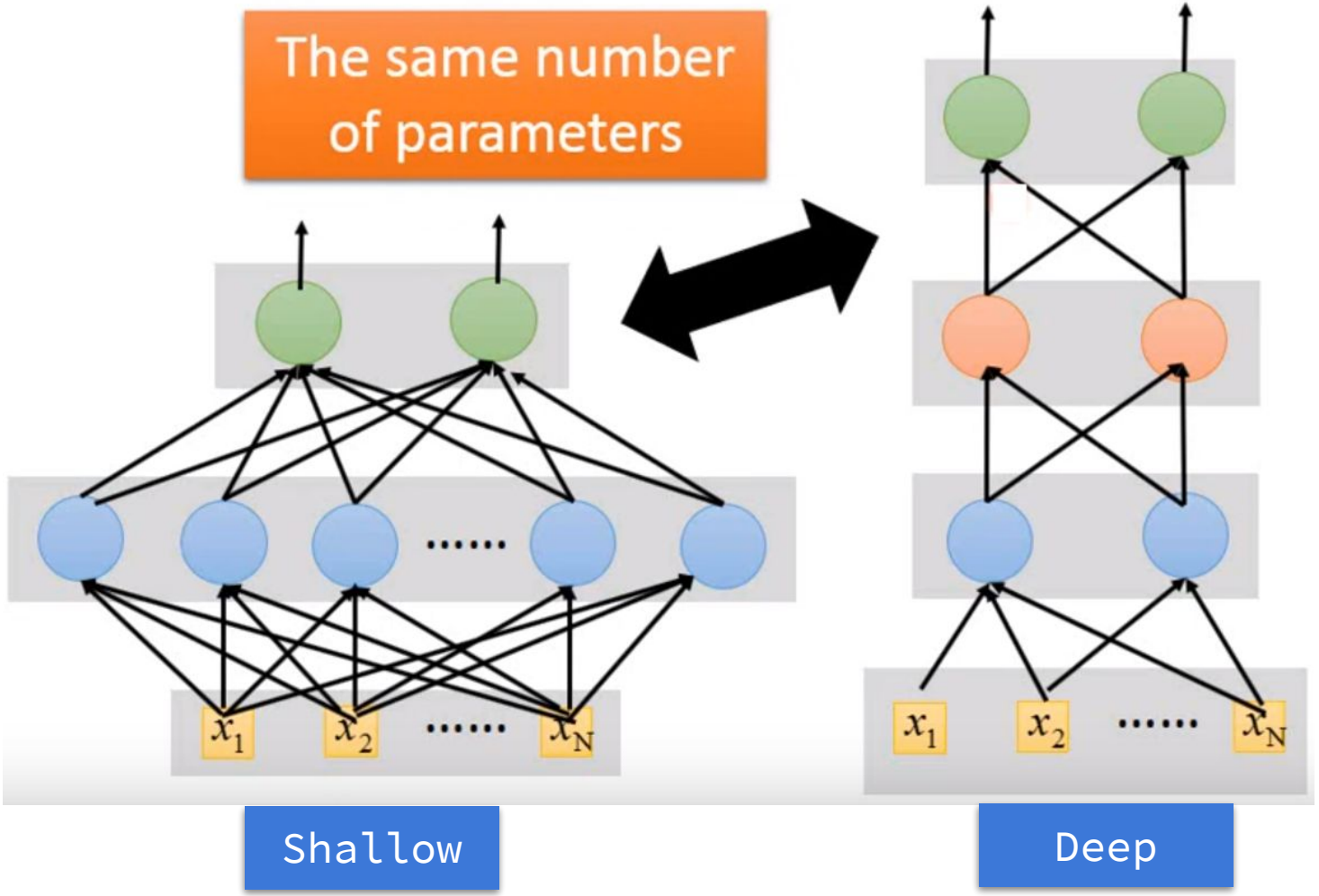
Vanishing Gradient Problem



Sigmoid activation funciton 已經證實有這個問題！

Network的架構

Fat + Short vs Thin + Tall



Network的架構 – 語音轉文字

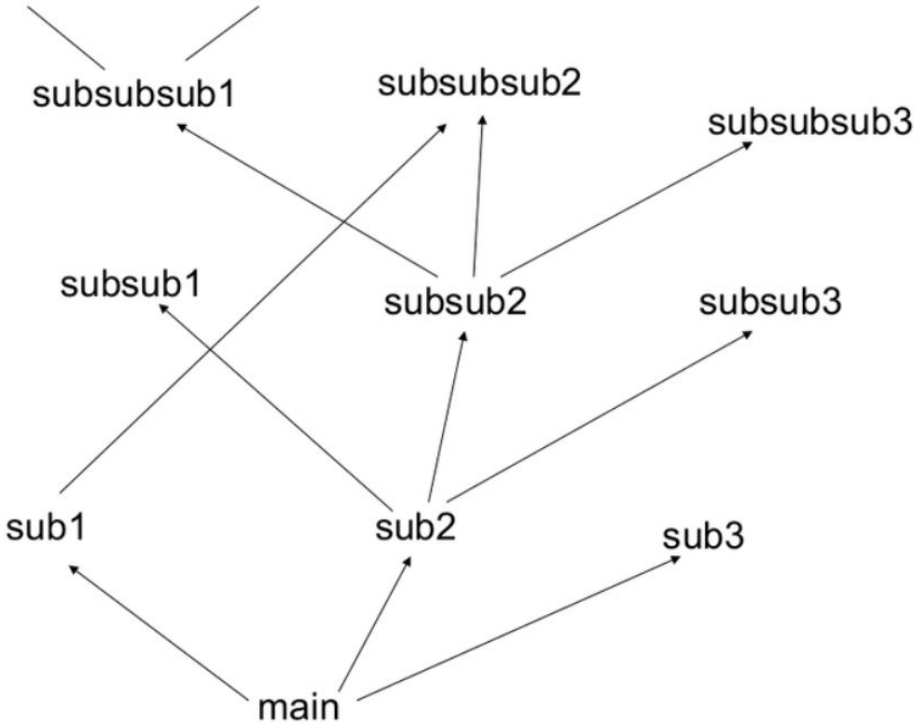
Layer X Size	Word Error Rate (%)	Layer X Size	Word Error Rate (%)
1 X 2k	24.2		
2 X 2k	20.4		
3 X 2k	18.4		
4 X 2k	17.8		
5 X 2k	17.2	1 X 3772	22.5
7 X 2k	17.1	1 X 4634	22.6
		1 X 16k	22.1

Seide, Frank, Gang Li, and Dong Yu. "Conversational Speech Transcription Using Context-Dependent Deep Neural Networks." Interspeech. 2011.

Network的架構

Deep Neural Network可以想成一種模組化的概念

Coding的概念:
不要在main重頭寫到尾!

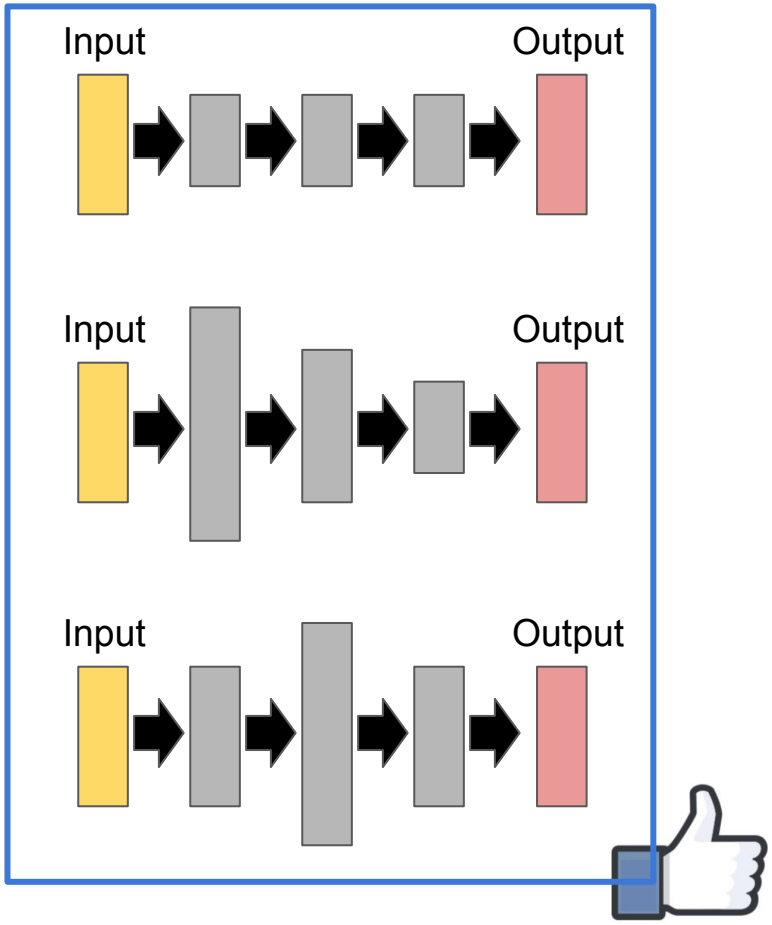


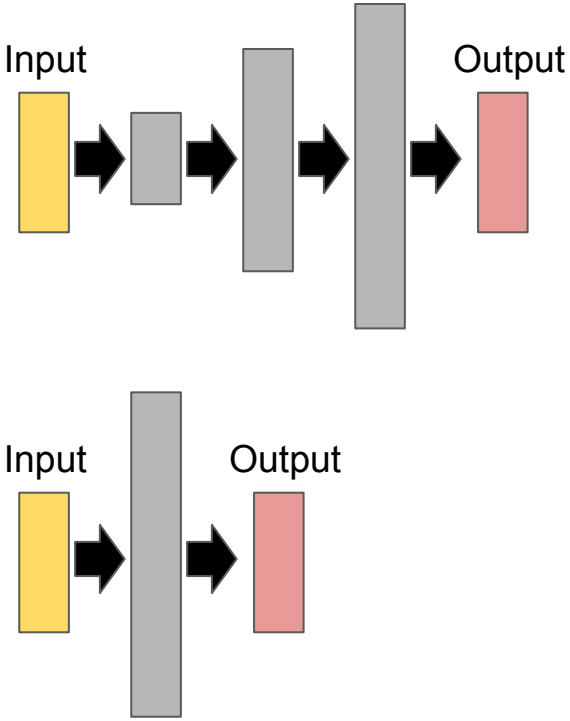
Network的架構

Modularization

這是Data Driven的模組化，你沒法決定第一每個Layer到底執行什麼功能，但是沒有模組化就沒有機會!

Network的架構


Input
Output
Input
Output
Input
Output

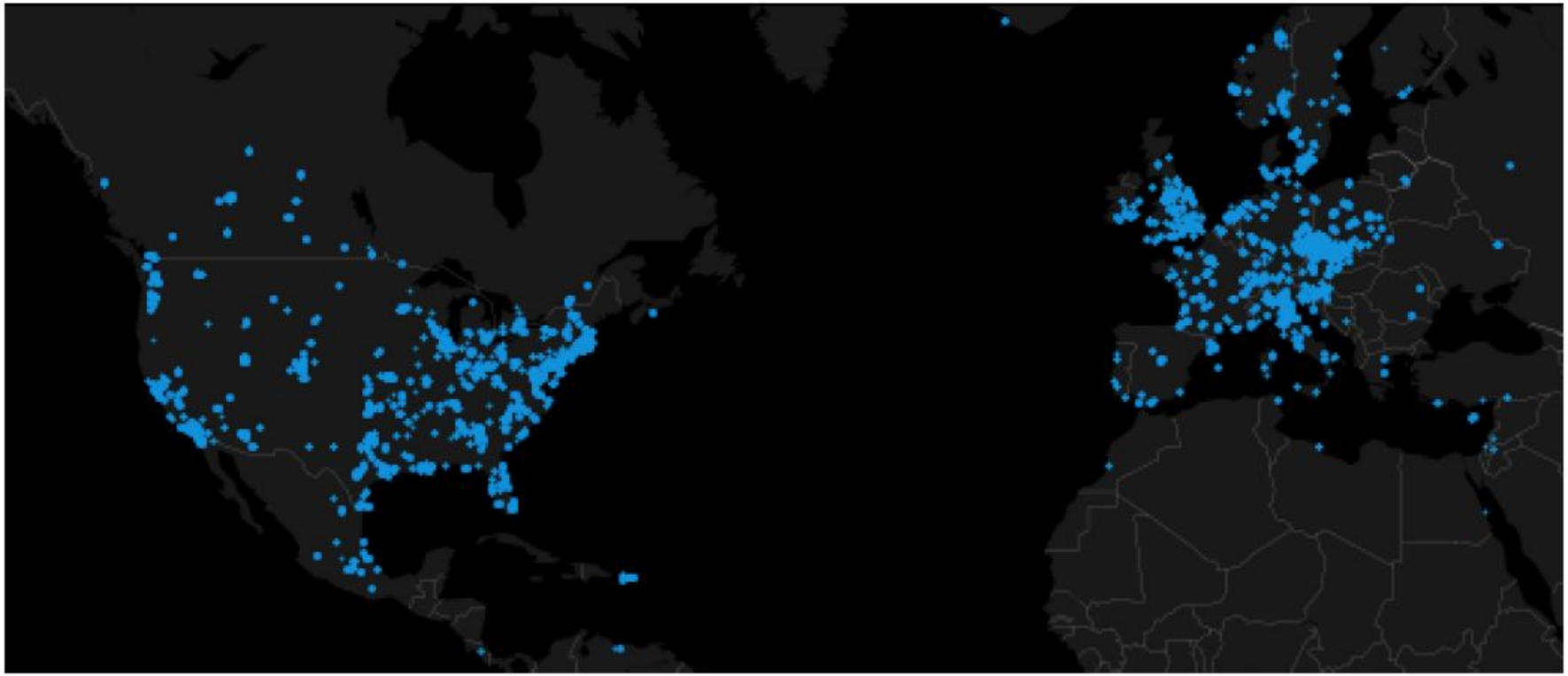

Input
Output
Input
Output

Deep Neural Network(DNN) Tips

Classification:
說太多了，動手實作吧！

直接來實作吧！

寶可夢過去出現的時間與地點紀錄 (dataset from Kaggle)



Ref: https://www.kaggle.com/kostyabahshetsyan/d/semioniy/predictemall/pokemon-geolocation-visualisations/notebook

範例資料: Raw Data Overview

latitude	longitude	appearedTimeOfDay	appearedYear	terrainType	closeToWater	city	weather	temperature	windSpeed	·····	cooc_151	class
20.525745	-97.460829	night	2016	14	0	Mexico_City	Foggy	25.5	4.79	·····	0	16
20.523695	-97.461167	night	2016	14	0	Mexico_City	Foggy	25.5	4.79	·····	0	133
38.90359	-77.19978	night	2016	13	0	New_York	Clear	24.2	4.29	·····	0	16
47.665903	-122.312561	night	2016	0	1	Los_Angeles	PartlyClou	15.6	5.84	·····	0	13
47.666454	-122.311628	night	2016	0	1	Los_Angeles	PartlyClou	15.6	5.84	·····	0	133
-31.95498	115.853609	night	2016	13	0	Perth	PartlyClou	16.5	6.39	·····	0	21
-31.954245	115.852038	night	2016	13	0	Perth	PartlyClou	16.5	6.4	·····	0	66
26.235257	-98.197591	night	2016	13	0	Chicago	Clear	28	11.26	·····	0	27
20.525554	-97.4588	night	2016	14	0	Mexico_City	Foggy	25.5	4.79	·····	0	35
32.928558	-84.340278	night	2016	8	0	New_York	Clear	23.7	3.94	·····	0	19
32.930646	-84.339867	night	2016	8	0	New_York	Clear	23.7	3.94	·····	0	116
32.943651	-84.334443	night	2016	8	0	New_York	Clear	23.7	3.94	·····	0	74
26.235552	-98.197249	night	2016	13	0	Chicago	Clear	28	11.26	·····	0	16
20.52577	-97.460237	night	2016	14	0	Mexico_City	Foggy	25.5	4.79	·····	0	19
26.236029	-98.196908	night	2016	13	0	Chicago	Clear	28	11.26	·····	0	19
47.664333	-122.312645	night	2016	0	1	Los_Angeles	PartlyClou	15.6	5.84	·····	0	19
20.526489	-97.460745	night	2016	14	0	Mexico_City	Foggy	25.5	4.79	·····	0	16
53.611417	-113.369528	night	2016	12	0	Edmonton	Clear	8.9	1.47	·····	0	13

問題: 會出現哪一隻神奇寶貝呢？

範例資料： 預測會出現哪一隻神奇寶貝?

- 時間:local.hour,local.month, DayofWeek…
- 天氣:temperature,windSpeed,pressure…
- 位置:longitude,latitude,pokestop…
- 環境:closeToWater,terrainType…
- 十分鐘前有無出現其他寶可夢
  - 例如:cooc_1=1十分鐘前出現過class=1隻寶可夢
- class 就是我們要預測目標

為了簡化複雜度，我們只挑出五種類別

Data Preprocessing Tips

- Neural network運算只接受數字(numeric)
- 如何處理非數值資料?
  - Categorical variables
  - Ordinal variables
- 不同feature的數值範圍差異是否會有影響?
  - 溫度: 0 ~ 40度
  - 距離: 0 ~ 10000公尺

Data Preprocessing Tips

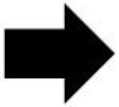
- Categorical variables
  - One Hot Encoding: 回想獵人的系別，強化系、變化系...
    - 一個類別一個function對付
    - 避免產生不必要的距離關係!
  - 特殊Case: 地址
    - 台中市大馬路無尾巷

    ⇒ 轉成經緯度{25.04,121.61}

Data Preprocessing Tips

- Ordinal variables(順序資料)
  - Def: 跟數值一樣有大小關係(順序關係)卻又像類別變數
  - For example: 衣服size, {S, M, L, XL}
    ⇒ {1, 2, 3, 4}
    ⇒ 當作類別變數One Hot Encoding
    S: [1, 0, 0, 0]
    M: [0, 1, 0, 0]
    …
  - Create a new feature using mean or median

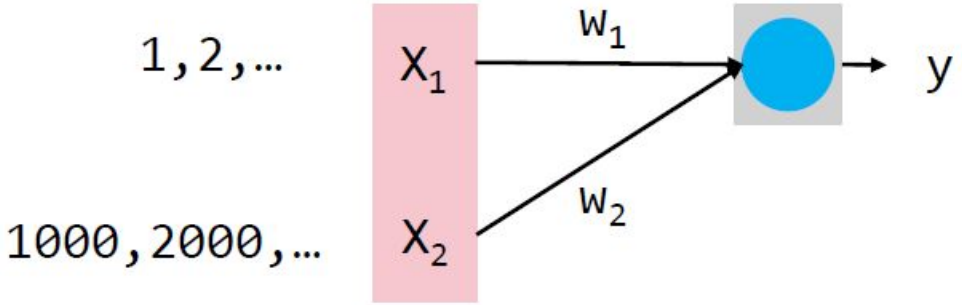
UID	Age
P1	0-17
P2	0-17
P3	55+
P4	26-35

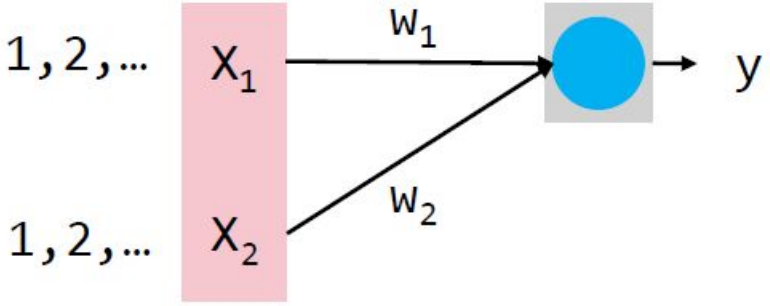


UID	Age
P1	15
P2	15
P3	70
P4	30

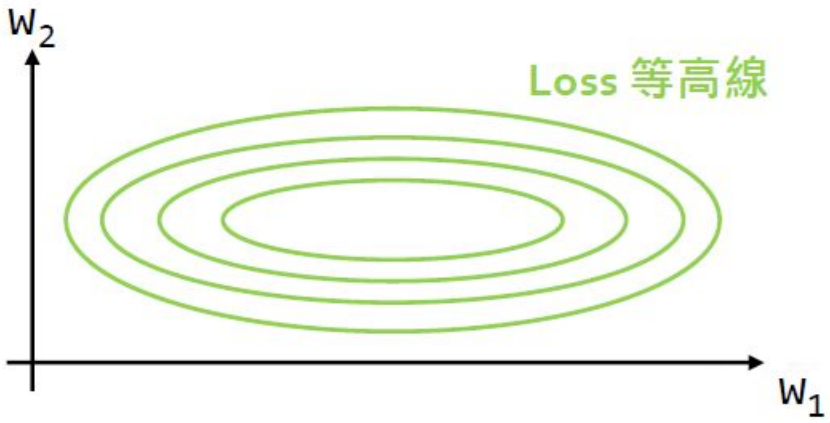
Data Preprocessing Tips

- 對於numeric欄位，處理不同數值範圍
  - 非常建議normalize，不要讓幾個變數主導整個function！





W_2 的修正(ΔW)對於 loss 的影響比較大



Data Preprocessing Tips

- Normalize methods
  - Min max scaler to [0, 1] $\frac{x_i - min}{max - min}$ (注意outlier)
  - Scale to standard normal distribution $\mu = 0,\ \sigma^2 = 1$ (Z-score standardize)
  - 以上兩種方式沒有guideline說什麼狀況下使用最好

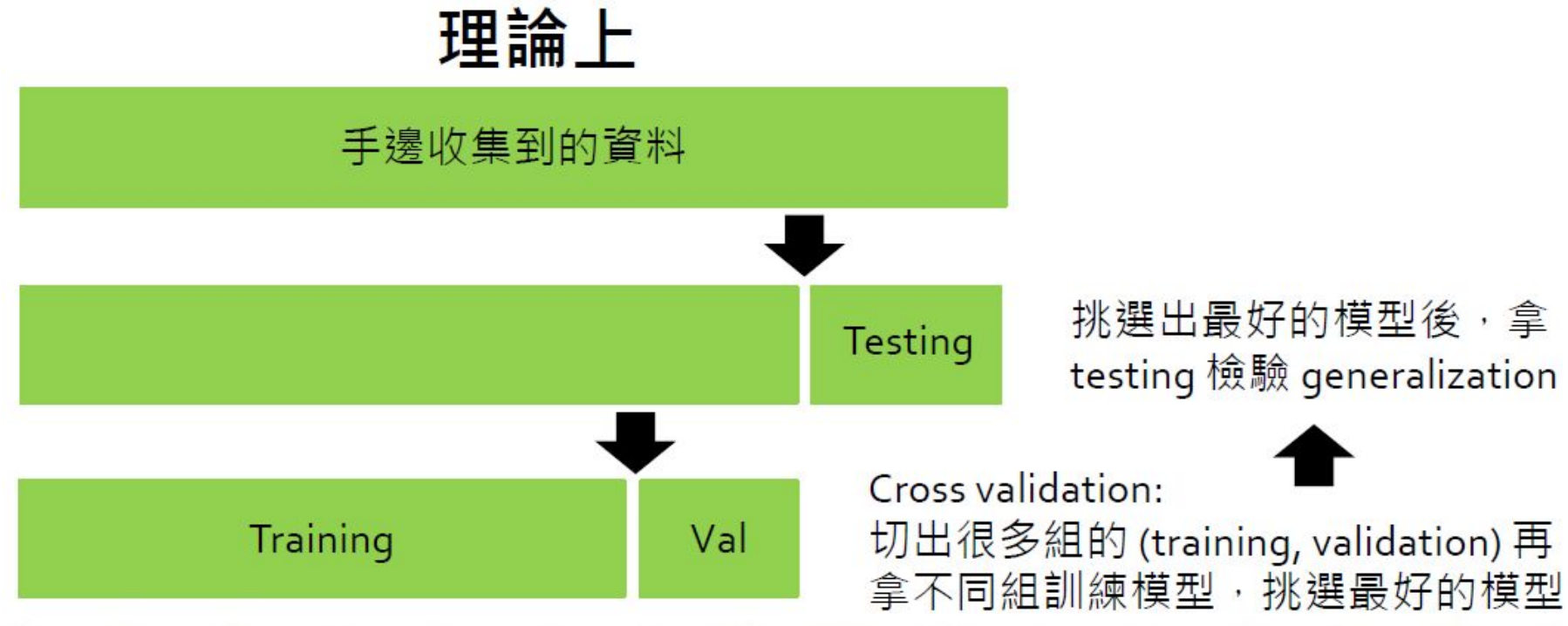
Deep Neural Network(DNN) Tips

寶可夢Deep Neural Network Tutorial

filename	tutorial_dnn_practice.ipynb
lab filename	**lab_tutorial_dnn_practice.ipynb**
input	200 features
label	5 class
neural network	DNN

How to Treat Your Data?

- Split train, valid, test data
  - Valid data用來挑選model
  - Test data檢驗模型的generalization
- **為簡化複雜度，Demo用traing + valid**



Data Preprocessing Tips

- 寶可夢資料已經處理好，只要load進來就好

```
''' The first column to the 199th column is used as input features '''
X_train = my_data[:,0:200]
X_train = X_train.astype('float32')

''' The 200-th column is the answer '''
y_train = my_data[:,200]
y_train = y_train.astype('int')

''' Convert to one-hot encoding '''
Y_train = tf.keras.utils.to_categorical(y_train,5)

''' Shuffle training data '''
from sklearn.utils import shuffle
X_train, Y_train = shuffle(X_train, Y_train, random_state=100)
```

input維度 = 200
(200個欄位)

label 5種類別
(one hot encoding)

Data Preprocessing Tips

- 使用data generate function ⇒ for iteration

Data Generator Function: Fetch Data Per Batch

```
def data_fn(X_train, Y_train, n_batch, shuffle=False):
    """train data iterator"""
    def fn():
        indices = utils.get_minibatches_idx(len(X_train), batch_size=n_batch, shuffle=shuffle)
        for ind in indices:
            yield X_train[ind], Y_train[ind]
    return fn
```

```
data (shape: (6, 200)):
[[ 0.0293 -0.5769  9.     ...,  0.     0.     0.    ]
 [ 0.0513 -0.6233  9.     ...,  0.     0.     0.    ]
 [ 0.1265 -0.5666  9.     ...,  0.     0.     0.    ]
 [ 0.0156 -0.6184  9.     ...,  0.     0.     0.    ]
 [-0.7685 -0.6619  9.     ...,  0.     0.     0.    ]
 [ 0.326  -0.5496  9.     ...,  0.     0.     0.    ]]

label (shape: (6, 5)):
[[ 0.  0.  0.  1.  0.]
 [ 0.  1.  0.  0.  0.]
 [ 0.  1.  0.  0.  0.]
 [ 0.  1.  0.  0.  0.]
 [ 0.  1.  0.  0.  0.]
 [ 0.  1.  0.  0.  0.]]
```

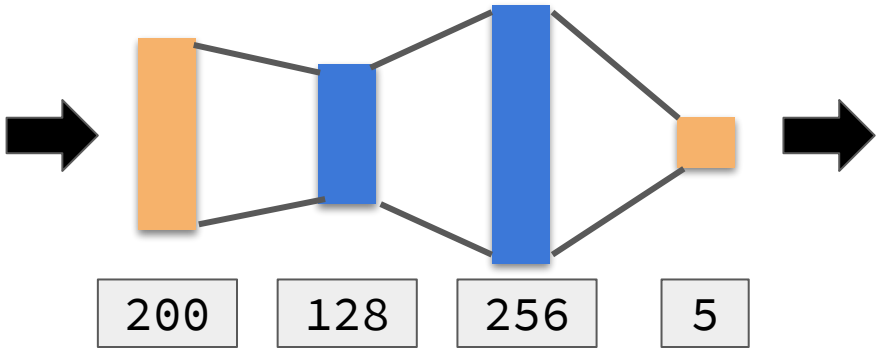
Before Build DNN

- BaseModel class(拿來繼承用的)
  - fit: training
    - train_fn: train data generator
    - valid_fn: valid data generator
    - n_epoch: number of epochs
    - lr: learning rate
    - callback: call on evrey epoch end
  - evaluate: calculate loss and accuracy
  - feed_dict: create data for placeholder
  - plot: plot accuracy and loss history

幫各位把routine的事情都做掉了，只要著重於Network架構的修改！

先訂一個模型架構跑跑看！

- 200(input)⇒ 128 ⇒ 256 ⇒ 5(output)



- Activation function: sigmoid
  - Output activation: softmax
- Loss function: mean squared error
- Learning rate = 0.01
- Batch size: 16 (每次train的筆數)
- Epoch: 30 (train完一次叫做一個epoch)

Tensorflow Code(Placeholder)

```
with tf.variable_scope("inputs"):
    self.inputs = tf.placeholder(tf.float32, [None, 200], name="inputs")
    self.labels = tf.placeholder(tf.float32, [None, 5], name="labels")
    self.is_train = tf.placeholder(tf.bool, None, name="is_train")
    # learning rate init value = 0.1
    self.lr = tf.Variable(0.1, trainable=False)
```

- inputs:
  - [None, 200] ⇒ None代表變動的batch size, 200 dim則是配合input dimesions
- labels:
  - [None, 5] ⇒ 5是配合output dimensions
- is_train指示是否是training狀態
  - ex: evaluate的時候不需要dropout!
- lr: learning rate

Tensorflow Code(DNN Structure)

```python
with tf.variable_scope("dnn"):
    # (Do!) 加入 hidden layer of 128 neurons
    nets = tf.layers.dense(self.inputs, 128, kernel_initializer=init_fn, activation=tf.nn.sigmoid, name="dnn1")
    # (Do!) 加入 hidden layer of 256 neurons
    nets = tf.layers.dense(nets, 256, kernel_initializer=init_fn, activation=tf.nn.sigmoid, name="dnn2")
    #
    nets = tf.layers.dense(nets, 5, kernel_initializer=init_fn, activation=None, name="dnn3")
    self.pred = tf.nn.softmax(nets, name="pred")
```

- 兩層[128, 256] hidden layer
  - tf.layers.dense已包含 activation(wx + b)
- kernel_initializer: 初始化weight的方式
  - Uniform distribution: tf.glorot_uniform_initializerzer
  - Normal distribution: tf.glorot_normal_initializer
- 分類問題output activation function: tf.nn.softmax

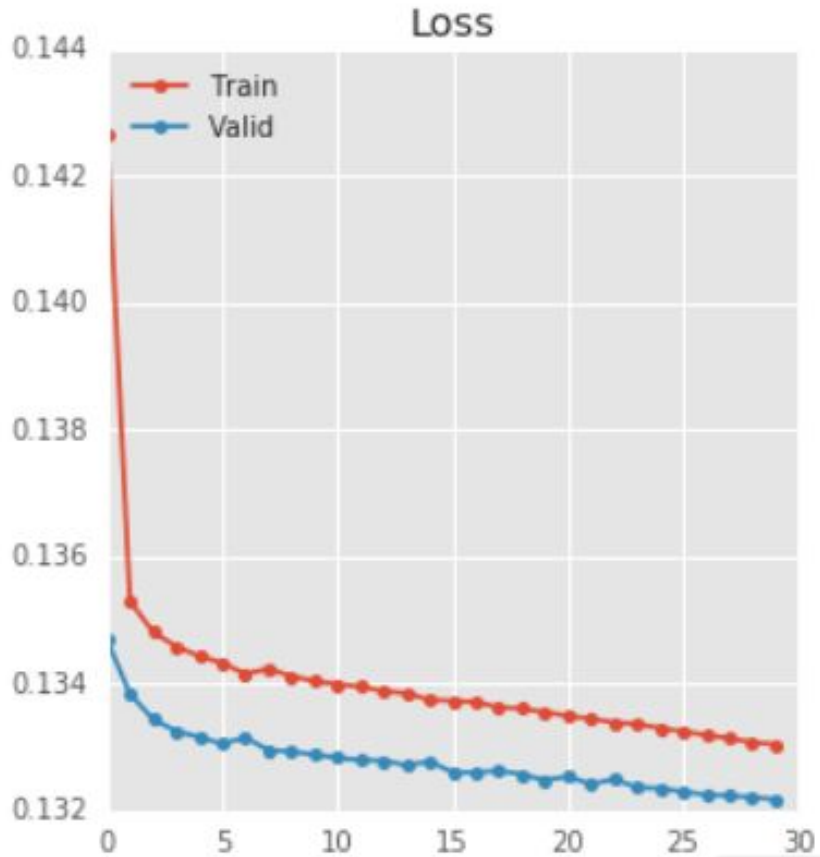
Tensorflow Code(Loss、Eval、Train)

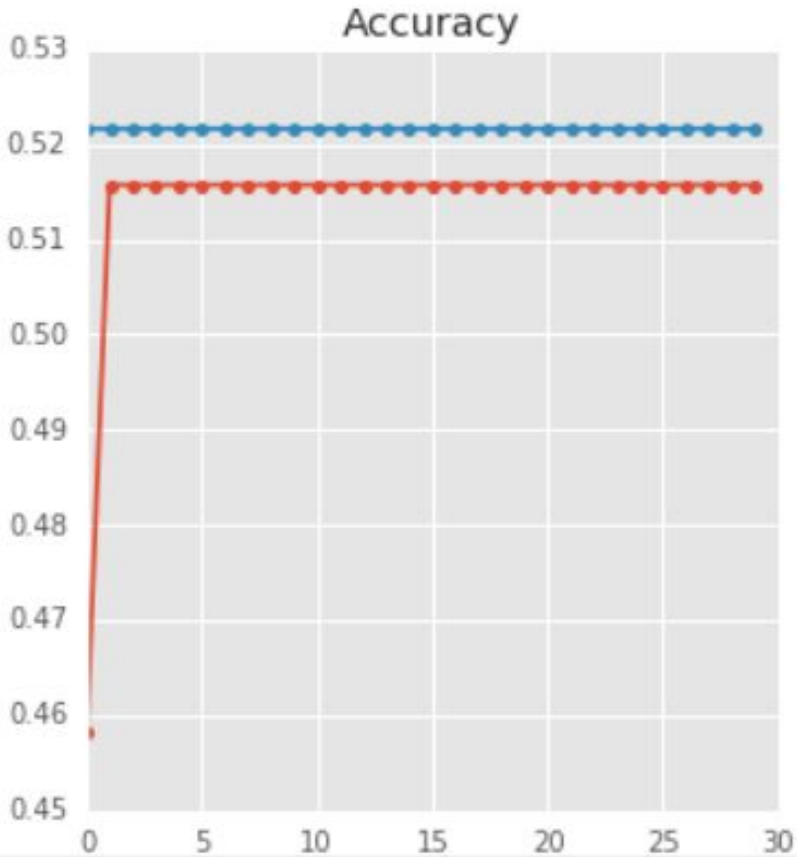
```
with tf.variable_scope("loss"):
    # 使用 mean squared error loss function
    self.loss = tf.losses.mean_squared_error(labels=self.labels, predictions=self.pred)

with tf.variable_scope("eval"):
    # 計算 accuracy
    self.acc = tf.reduce_mean( tf.to_float(tf.equal(tf.argmax(self.pred, 1), tf.argmax(self.labels, 1))) )

with tf.variable_scope("train"):
    self.train_op = tf.train.GradientDescentOptimizer(self.lr).minimize(self.loss)
```

- loss/loss function: tf.losses.mean_squared_error
- eval/accuracy:
  - pred: network output with softmax
  - label: the answer
  - 比較pred and label最大值的index ⇒ argmax
- train/optimizer:
  - 基本款tf.train.GradientDescentOptimizer

Result of ModelMSE

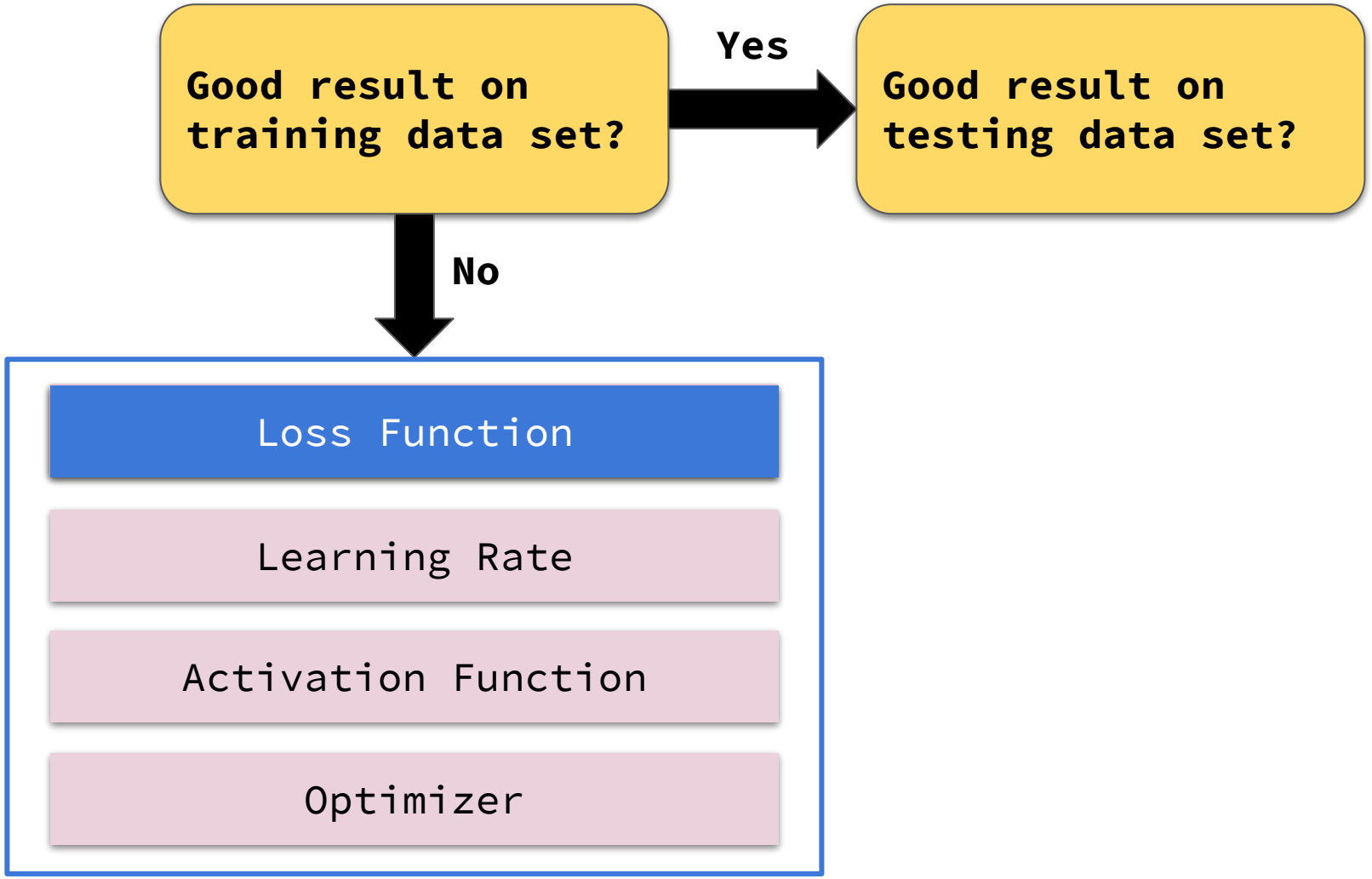
```
mean square error model!
30/30.  train loss: 0.133, valid loss: 0.132, train acc: 0.516, valid acc: 0.522
```





Accuracy about 0.5, 好像不太妙啊...

Deep Neural Network(DNN) Tips



Lab: lab_dnn_tutorial_practice.ipynb (20 minutes)

lab filename	lab_tutorial_dnn_practice.ipynb
target	將loss function由MSE修改成cross entropy

Classification With Cross Entropy

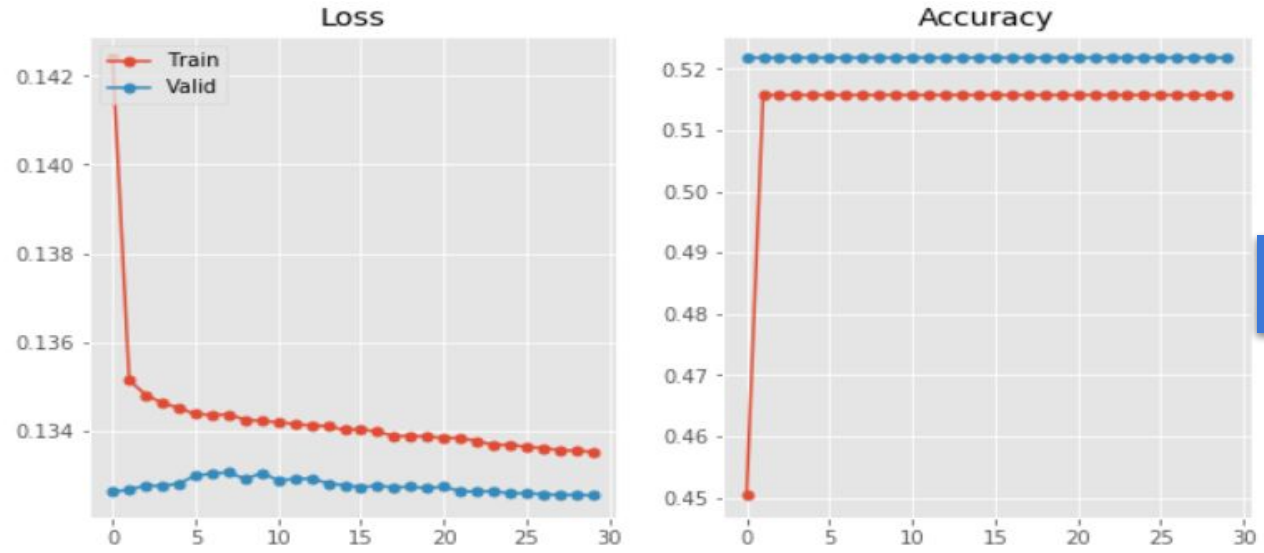
+ learning_rate: 0.01
+ loss function: softmax cross entropy
+ optimizer: 基本款 => GradientDescentOptimizer
+ activation function: sigmoid

Tensorflow Code(Loss Function)

```
with tf.variable_scope("loss"):
    # loss function 改用 cross entropy
    self.loss = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(labels=self.labels, logits=nets))
```

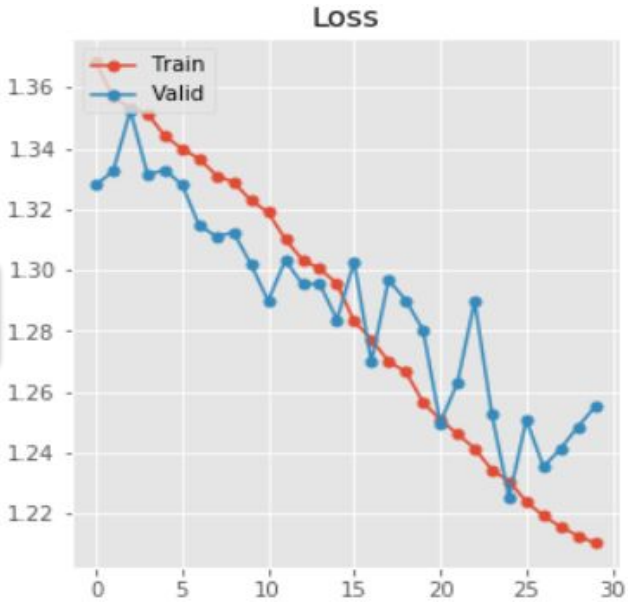
- loss function: tf.nn.softmax_cross_entropy_with_logits
  - **這個function會自動幫你的prediction加上softmax**

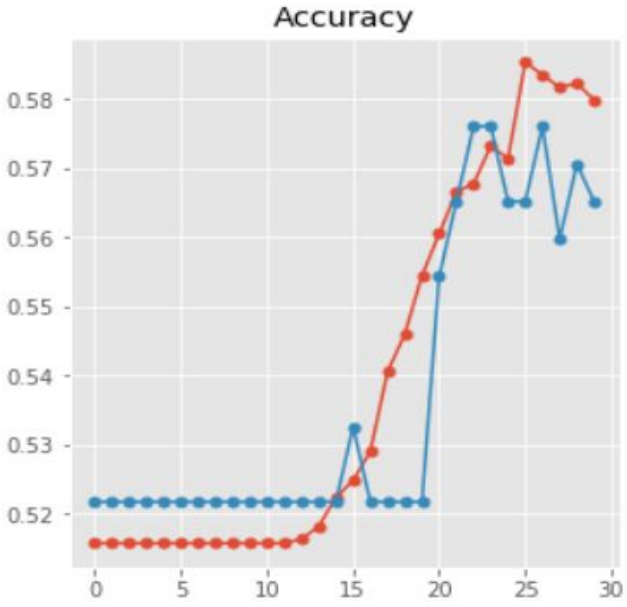
Result - MSE vs Cross Entropy



MSE

Cross Entropy





還記得Classification的微分推導嗎?

分類使用Cross entropy導數

$$-\left(\hat{y}^n - f_{w,b}(x^n)\right)x_i^n$$

$\hat{y}^n = 1$ If $f_{w,b}(x^n) = 1$ ➡ 0

If $f_{w,b}(x^n) = 0$ ➡ $-x_i$

Large diff, large update

分類使用MSE導數

$$\left(f_{w,b}(x) - \hat{y}\right) f_{w,b}(x)\left(1 - f_{w,b}(x)\right)x_i$$

$\hat{y}^n = 1$ If $f_{w,b}(x^n) = 1$ ➡ 0

If $f_{w,b}(x^n) = 0$ ➡ 0

no update

How to Select Loss function

- Classification常用cross entropy
  - 搭配softmax當作output layer的activation
- Regression常用mean squared/absolute error

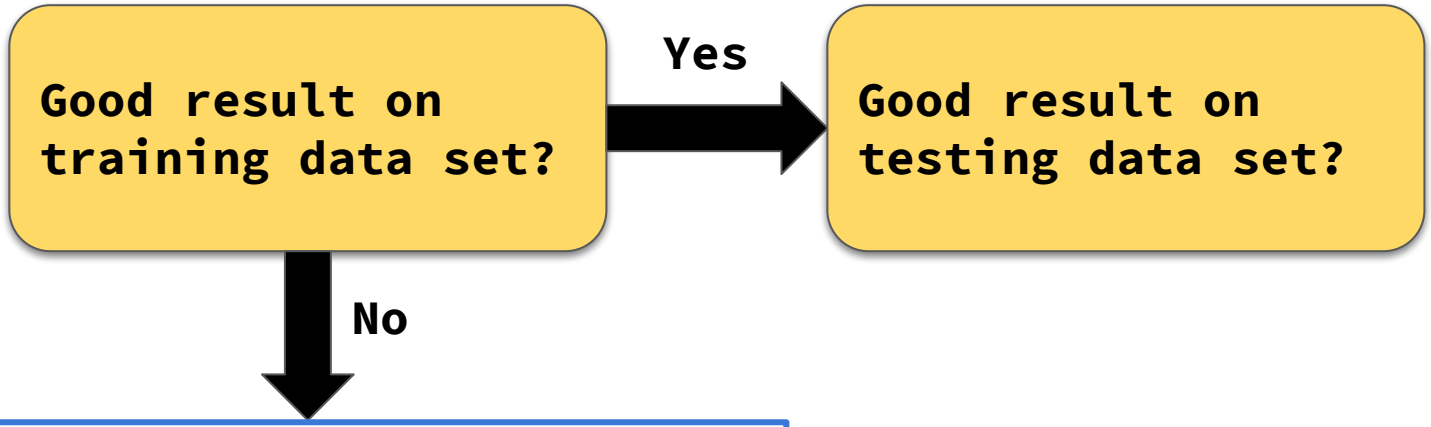
Lab: lab_dnn_tutorial_practice.ipynb (5 ~ 8 minutes)

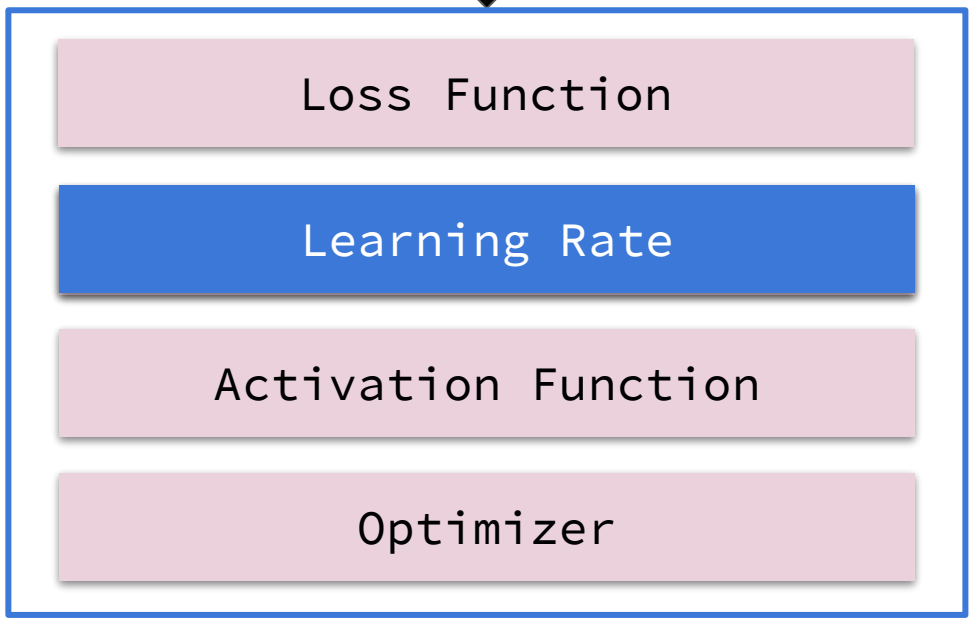
lab filename	lab_tutorial_dnn_practice.ipynb
target	嘗試不同的learning rate: [0.1, 0.01, 0.001]

Try Learning Rate!

+ learning_rate: 0.1, 0.01, 0.001
+ loss function: softmax cross entropy
+ optimizer: 基本款 => GradientDescentOptimizer
+ activation function: sigmoid

Tune Your Learning Rate





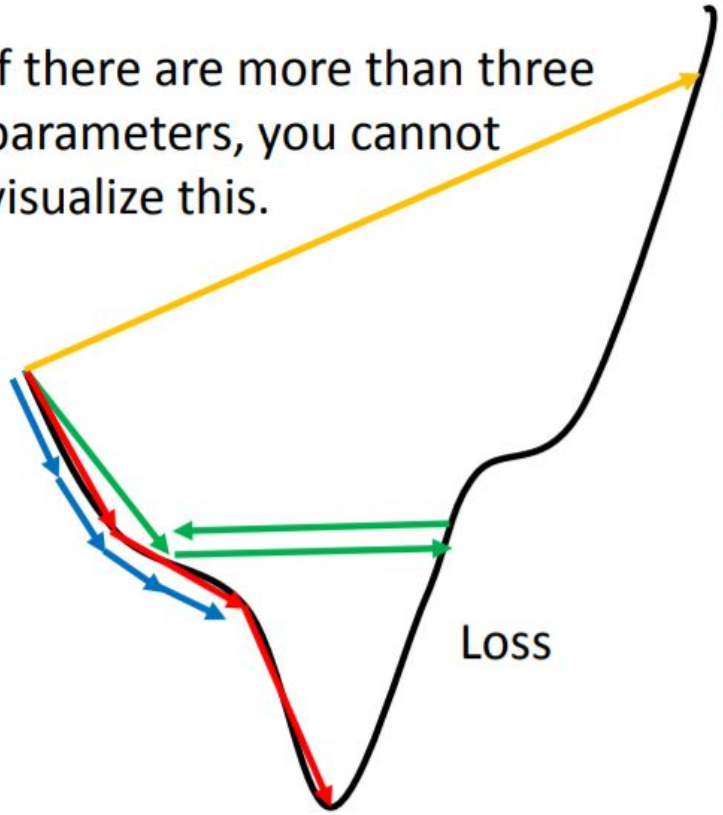
Tune Your Learning Rate

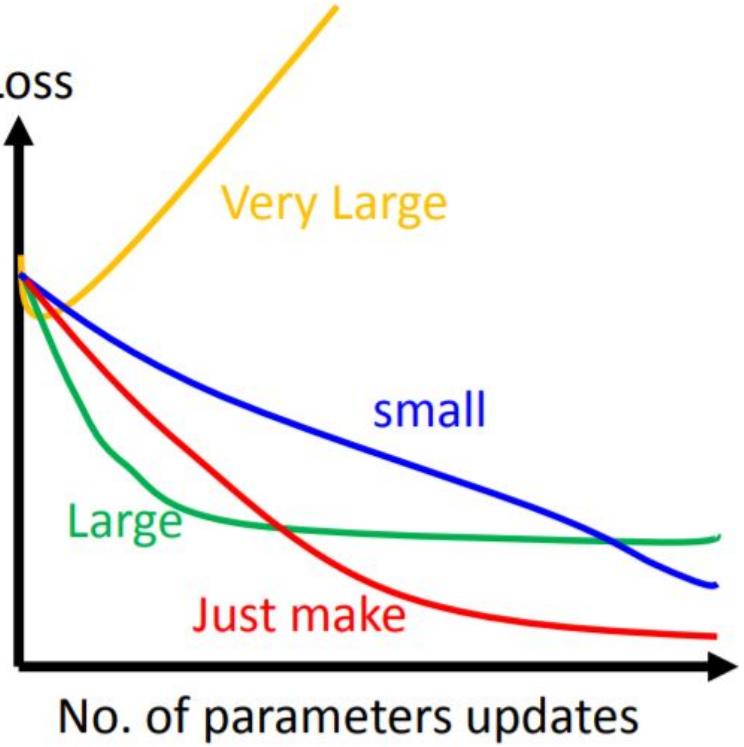
Learning Rate

$$\theta^i = \theta^{i-1} - \eta \nabla L\left(\theta^{i-1}\right)$$

Set the learning rate η carefully

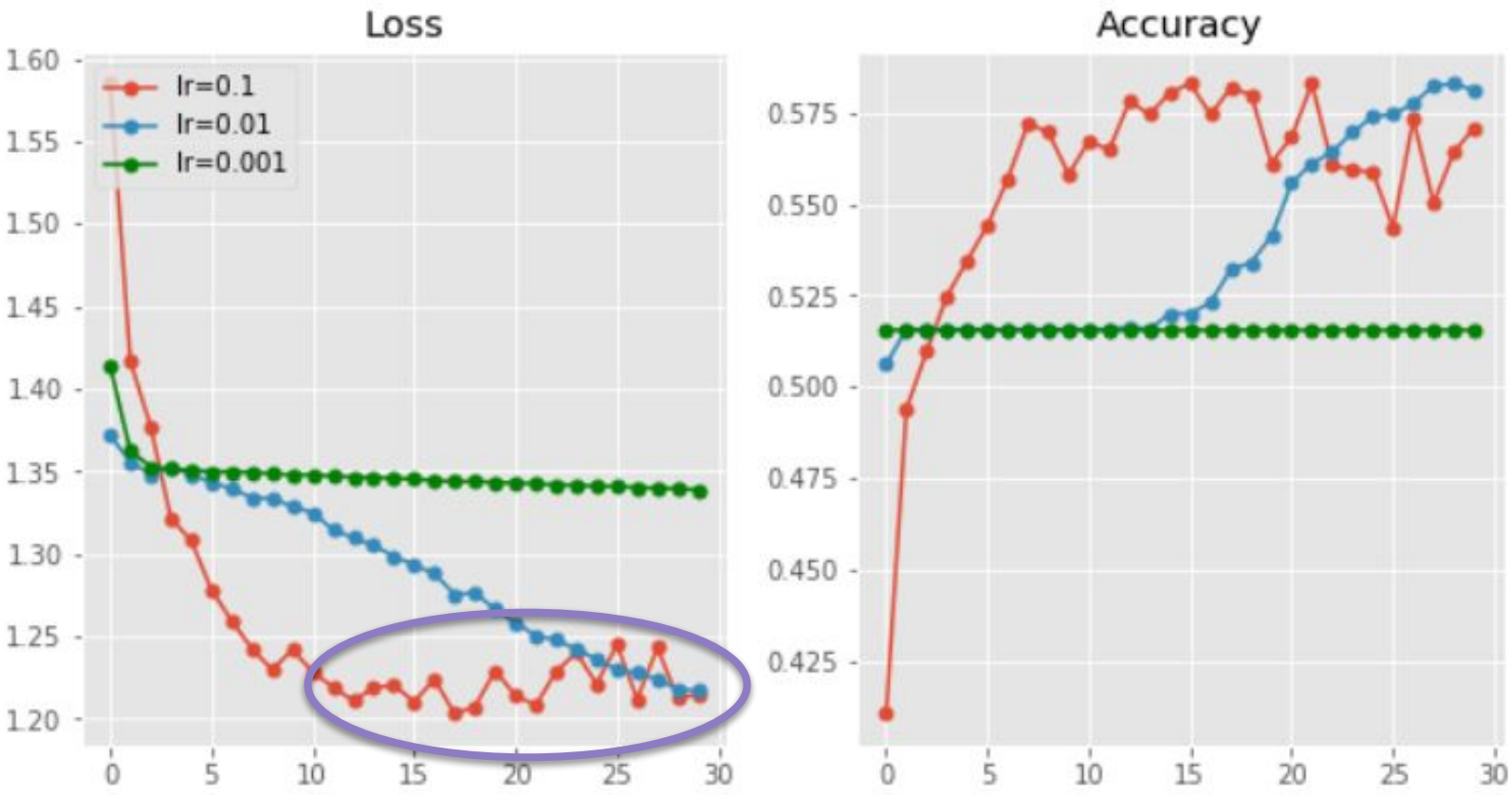
If there are more than three parameters, you cannot visualize this.





But you can always visualize this.

Tune Your Learning Rate

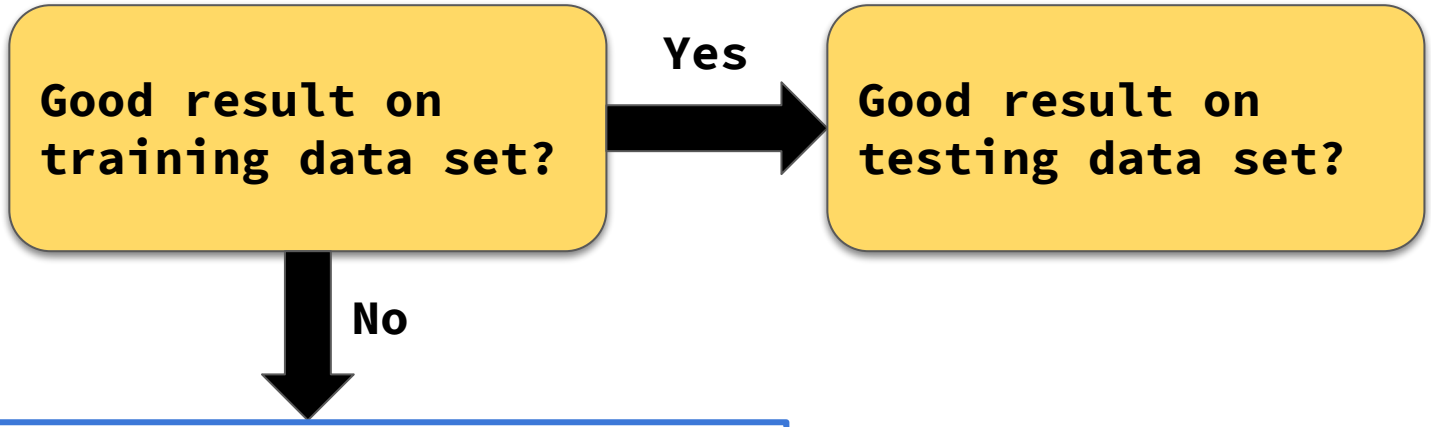


觀察loss，這樣的震盪比表示learning rate可能太大，
以這個狀況，0.01是最好的選擇

Tune Your Learning Rate

- 大多要試試看才知道，通常不會大於0.1
- 調整的步伐大概是一次5 ~ 10倍
  - 0.1 ⇒ 0.01 ⇒ 0.001
  - 0.001 ⇒ 0.01 ⇒ 0.1 ⇒ 0.5
  - ~~0.01 ⇒ 0.012 ⇒ 0.015 ⇒ 最高0.87分不能再高~~

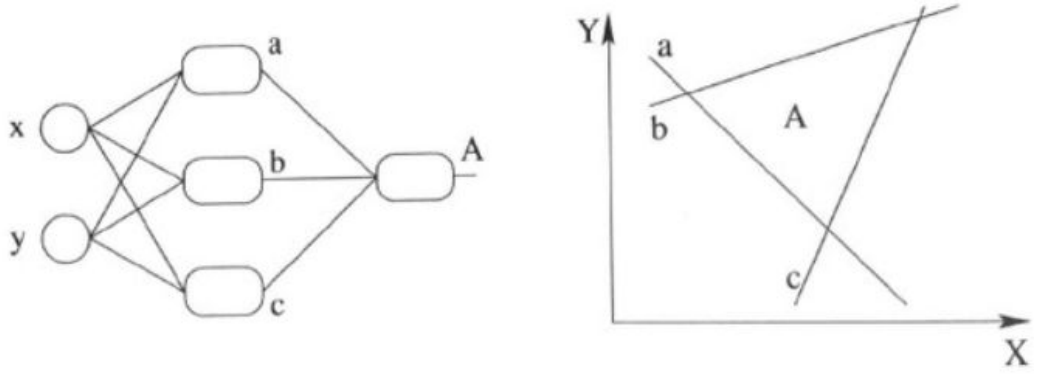
Choose Your Activation Function



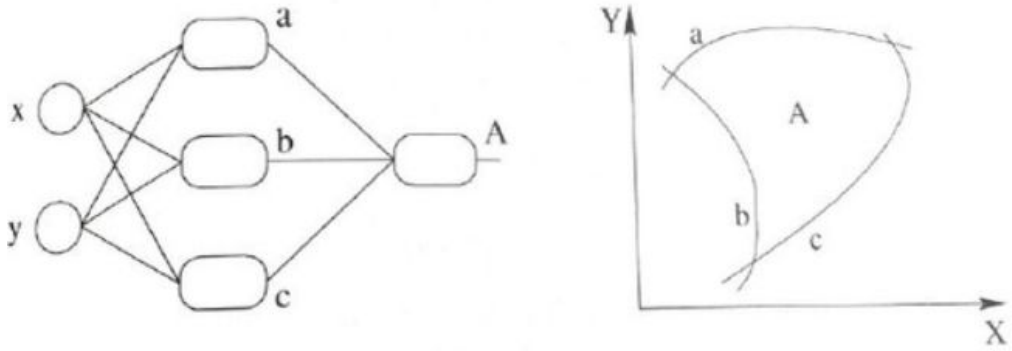
- Loss Function
- Learning Rate
- **Activation Function**
- Optimizer

Activation Function的初衷

- 把你的Neural network變成non-linear，加強fit能力



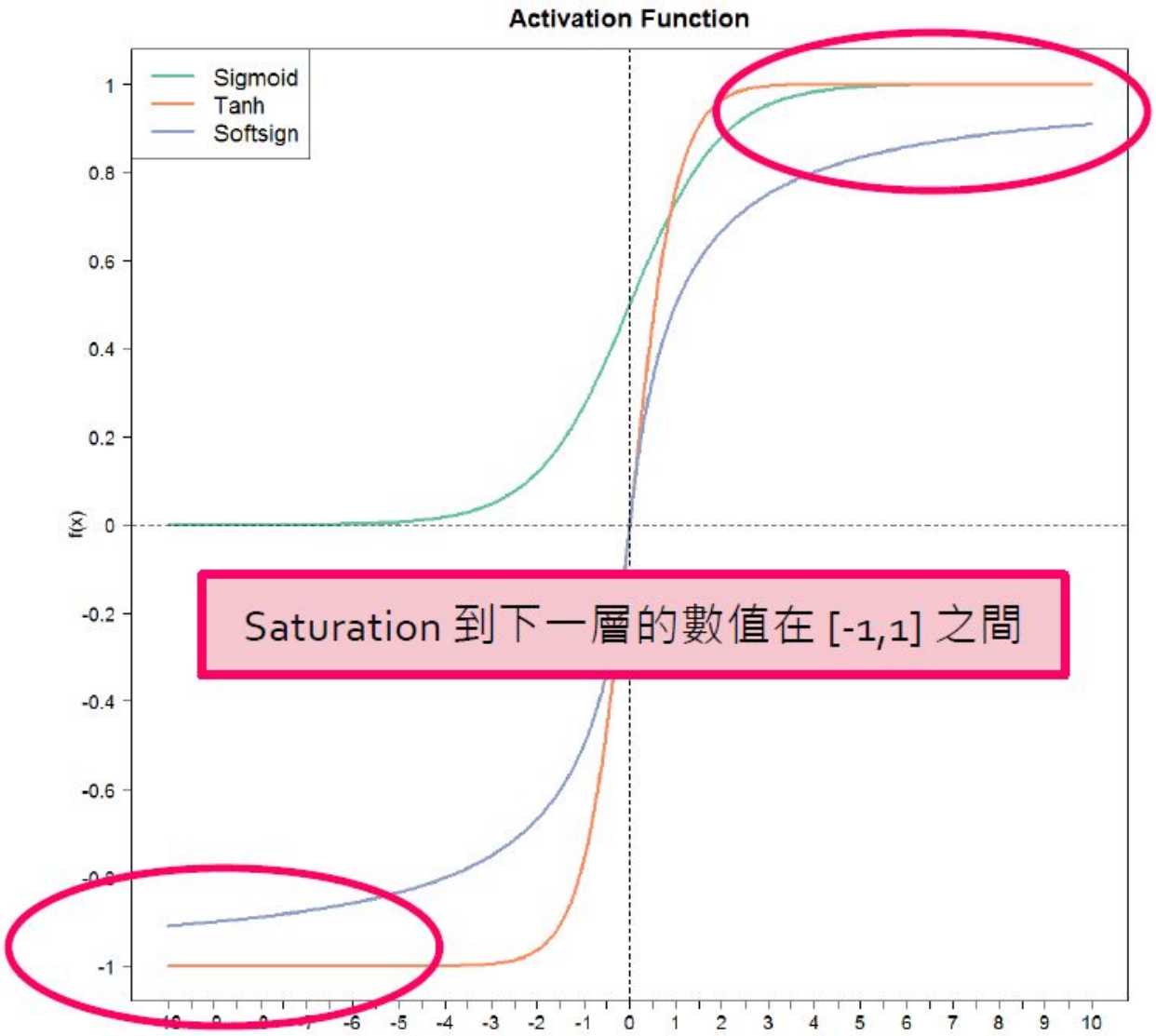
with step activation function



with sigmoid activation function

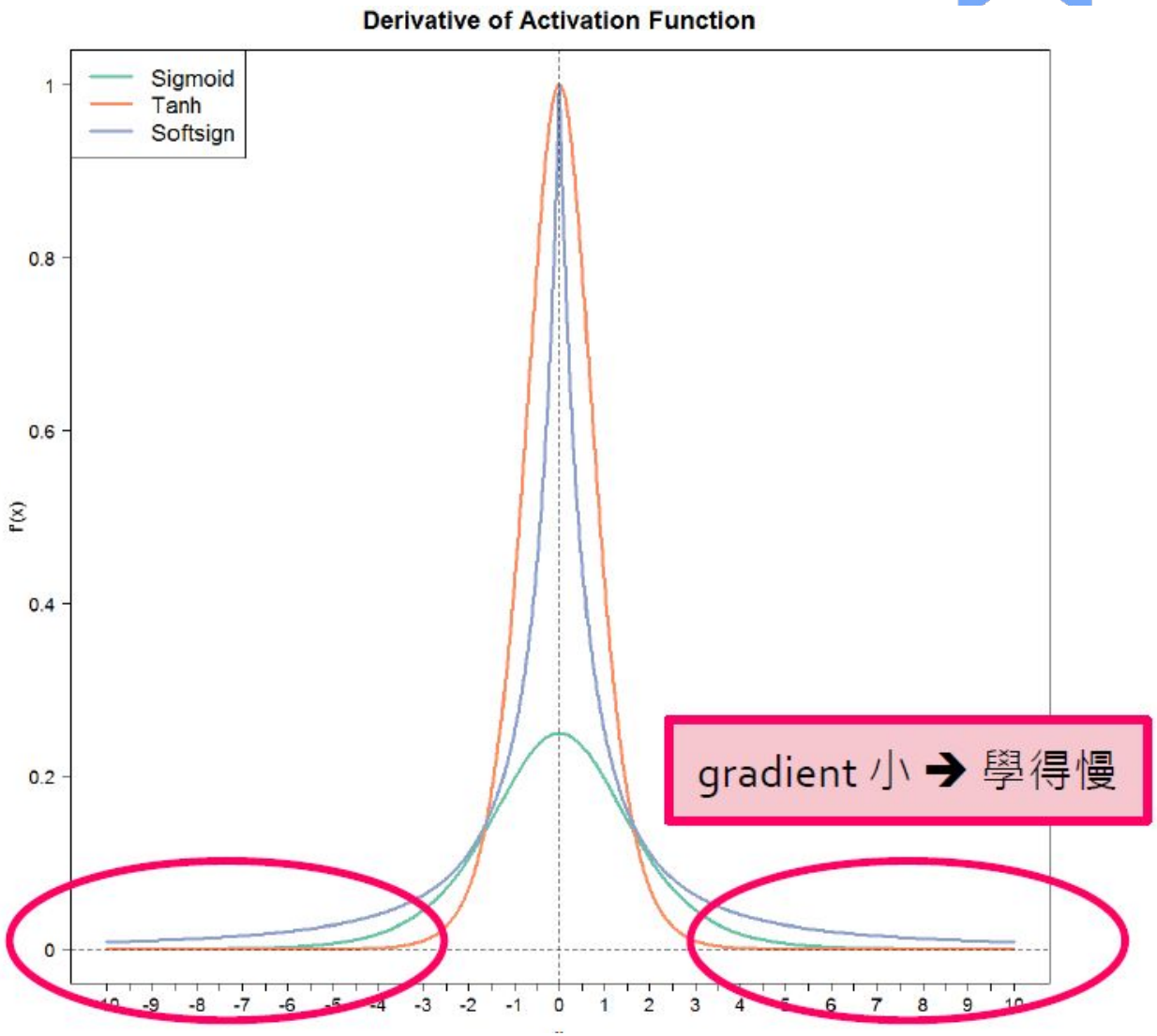
傳統的Activation Function

- Sigmoid
  - $f(x)=\frac{1}{(1+e^{-x})}$
- Tanh
  - $f(x)=\frac{(1-e^{-2x})}{(1+e^{-2x})}$
- Softsign
  - $f(x)=\frac{x}{(1+|x|)}$



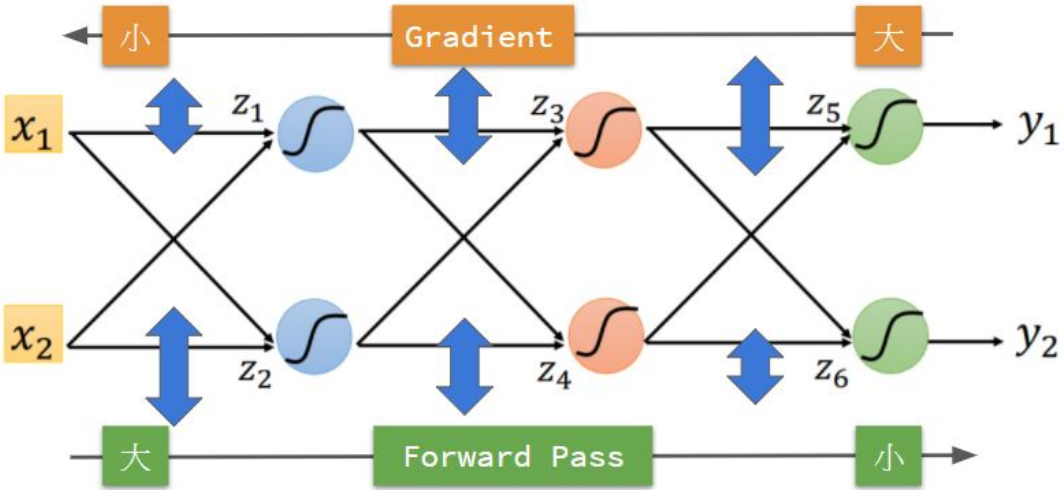
傳統的Activation Function

- Sigmoid
  - $df/dx = \frac{e^{-x}}{(1+e^{-x})^2}$
- Tanh
  - $df/dx = 1-f(x)^2$
- Softsign
  - $df/dx = \frac{1}{(1+|x|)^2}$



傳統的Activation Function

- Vanishing gradient problem
  - 原因: 每一層的input被壓縮到一個相對很小的output range

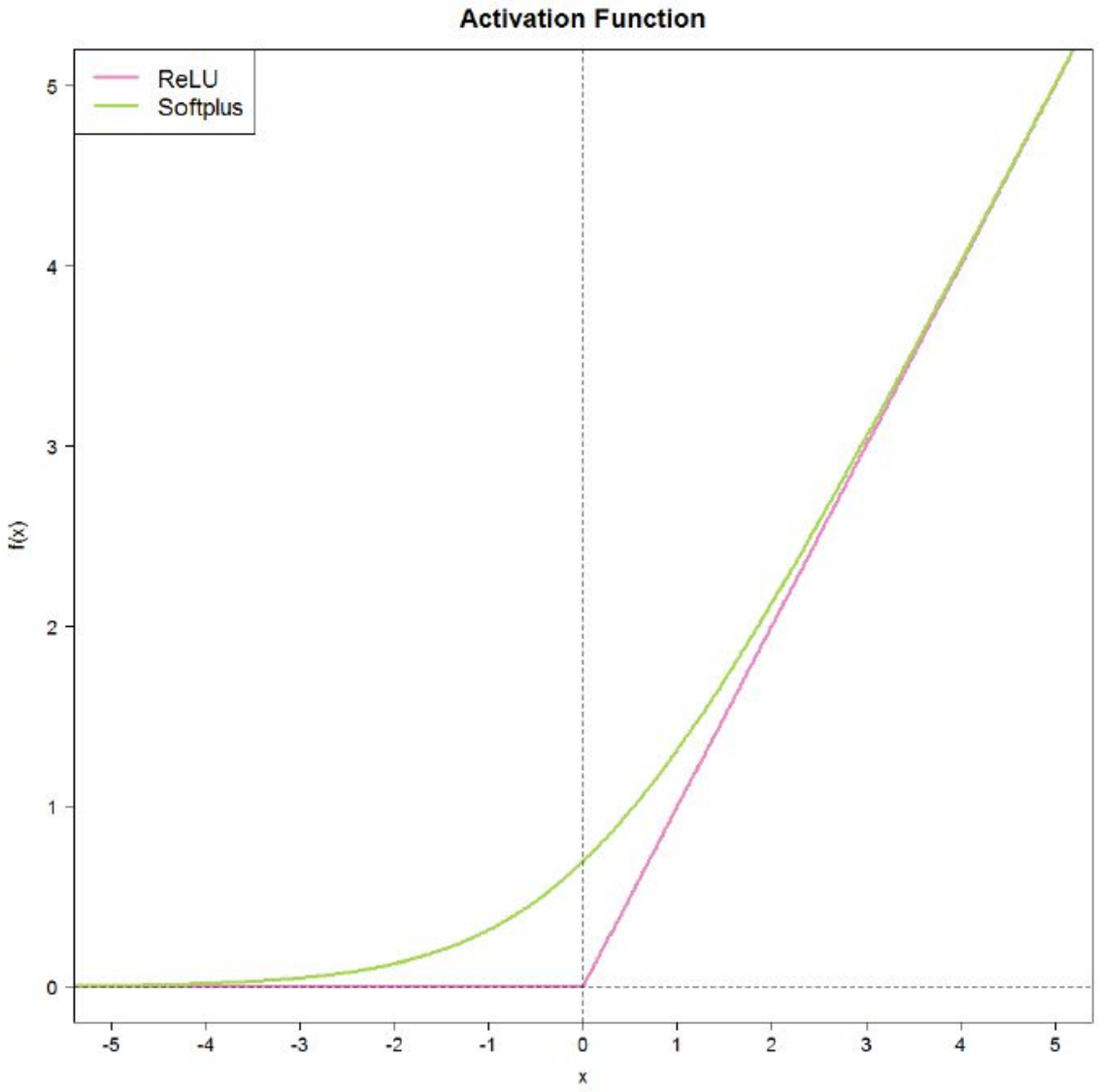


  - 結果: Gradient小無法有效地學習
  - Sigmoid, Tanh, Softsign都有這樣的特性

- **特別不適用於深的深度學習模型**

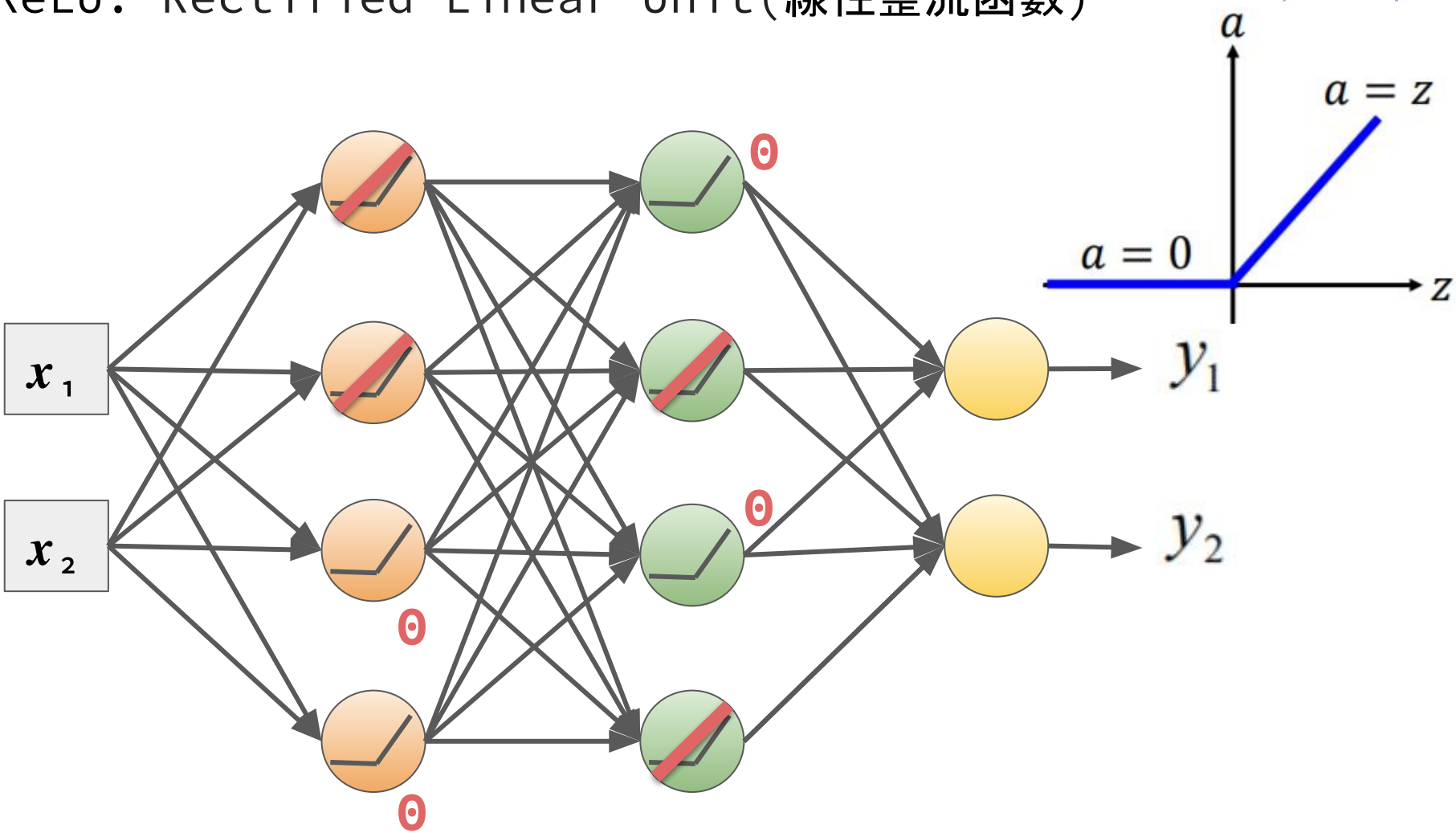
比較潮的Activation Function

- ReLU
  - f(x)=max(0,x)
  - df/dx=1 if x > 0,
    0 otherwise.
- Softplus
  - $f(x) = \ln(1+e^x)$
  - $df/dx = e^x/(1+e^x)$



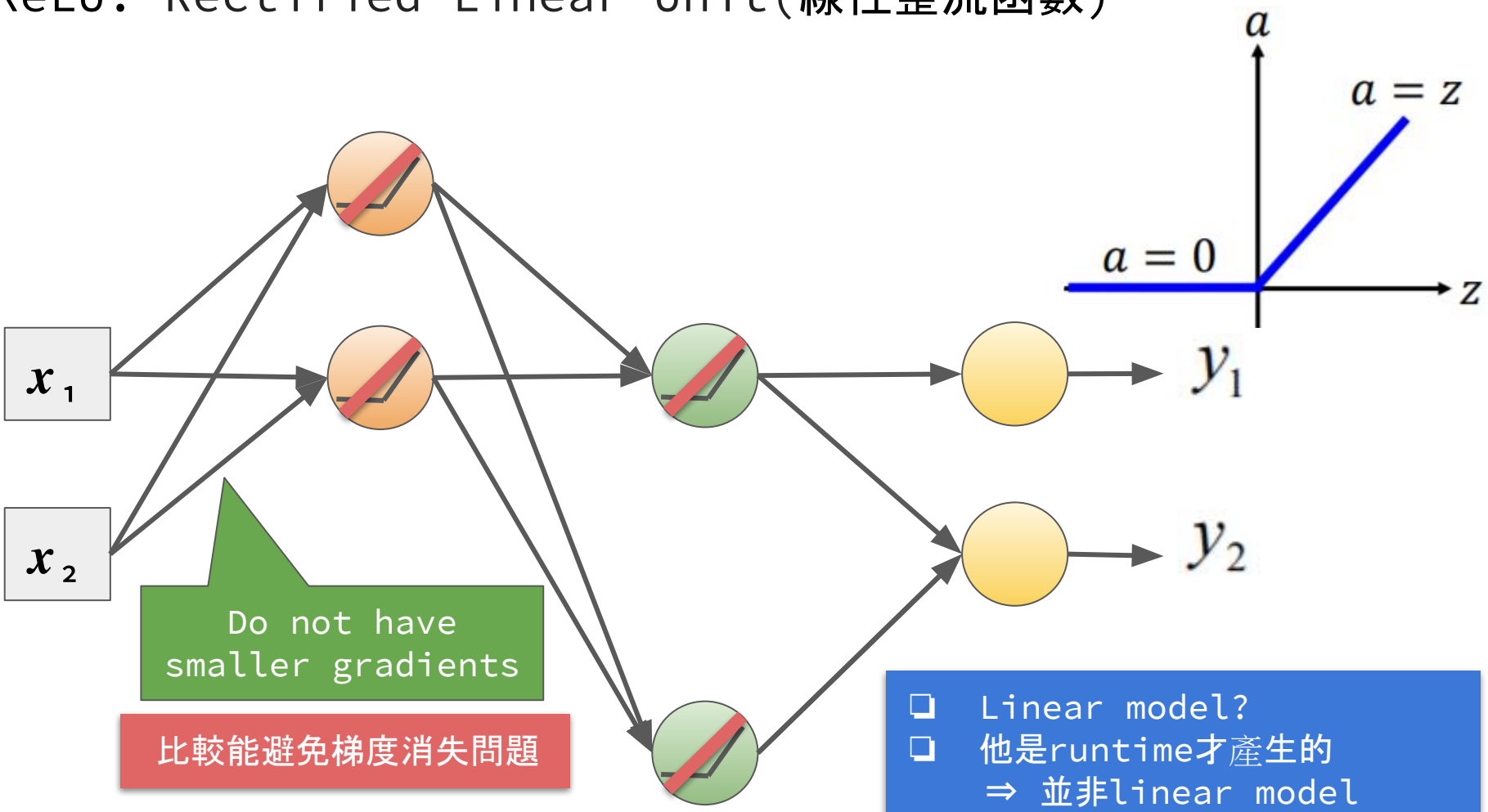
比較潮的Activation Function

ReLU: Rectified Linear Unit(線性整流函數)



比較潮的Activation Function

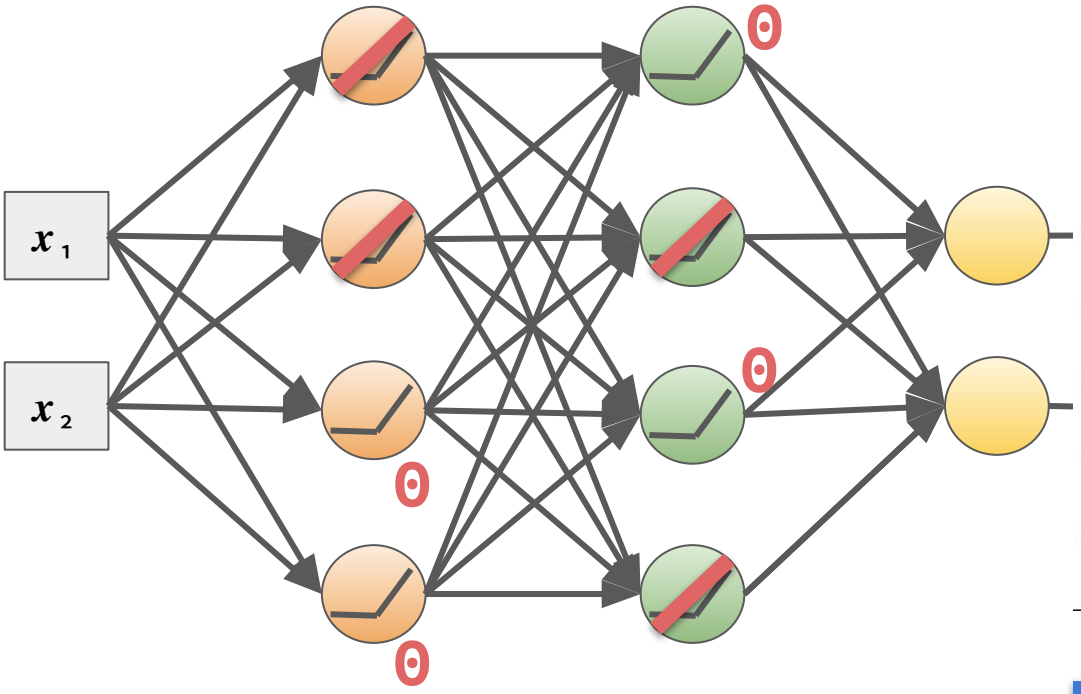
ReLU: Rectified Linear Unit(線性整流函數)

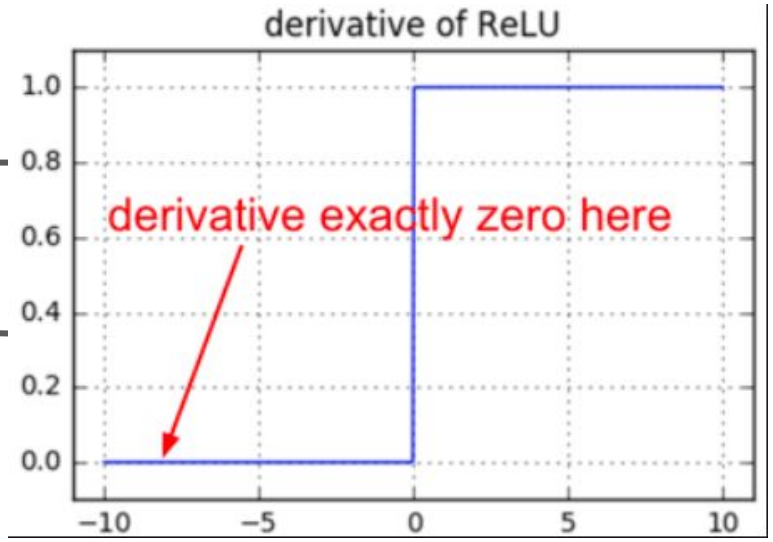


- Linear model?
- 他是runtime才產生的
  ⇒ 並非linear model
- Sparsity, 計算速度快

比較潮的Activation Function

ReLU: 還是有缺點...

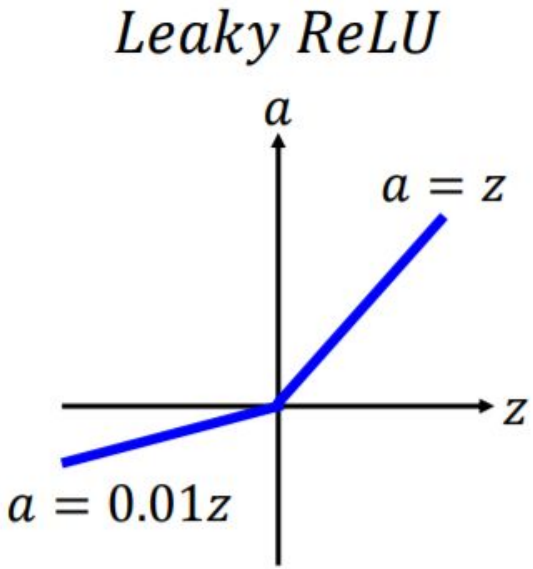


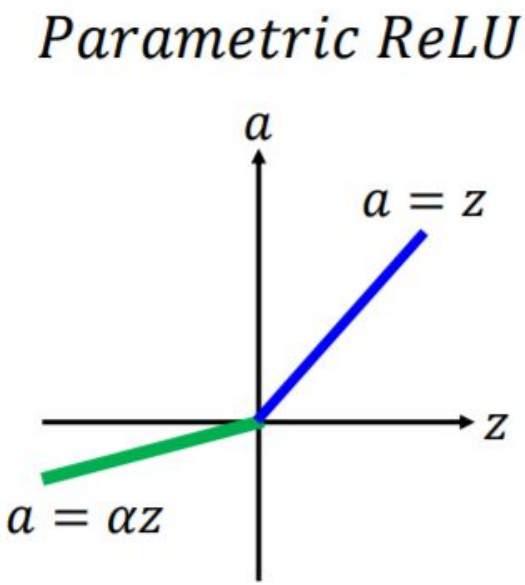


❑ Dying ReLU problem

比較潮的Activation Function

ReLU - variant





實際上Dying ReLU problem:

- ❑ 適當的weight init
- ❑ 適當的learning rate

α also learned by gradient descent

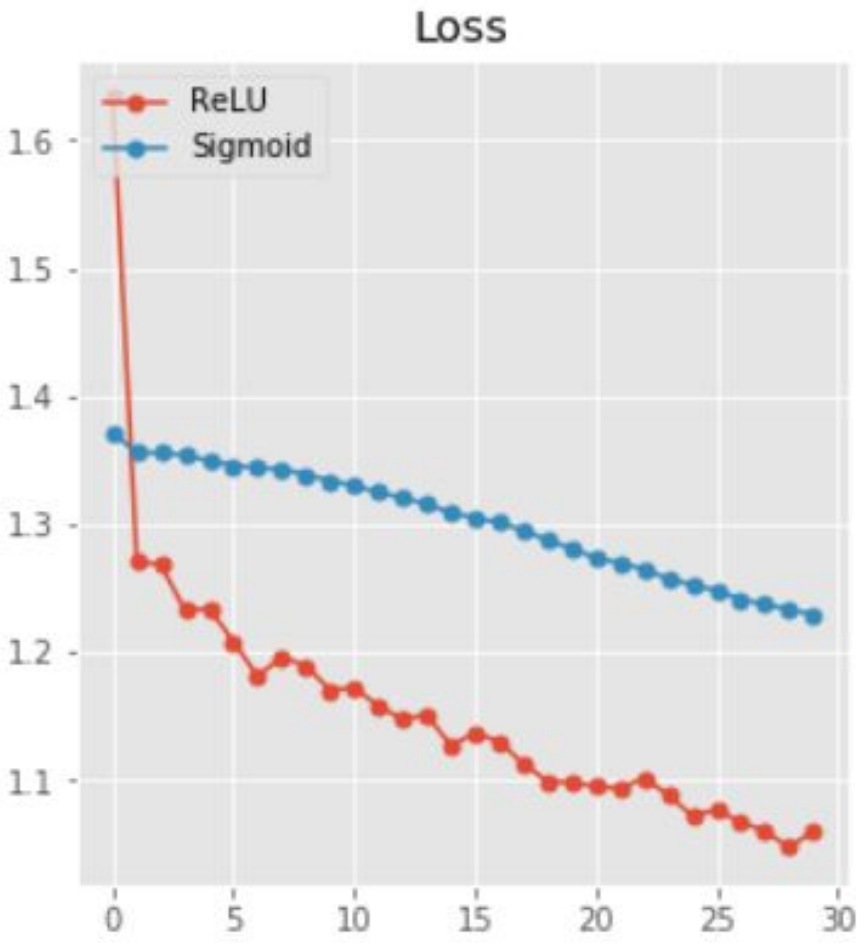
Lab: lab_dnn_tutorial_practice.ipynb (5 ~ 8 minutes)

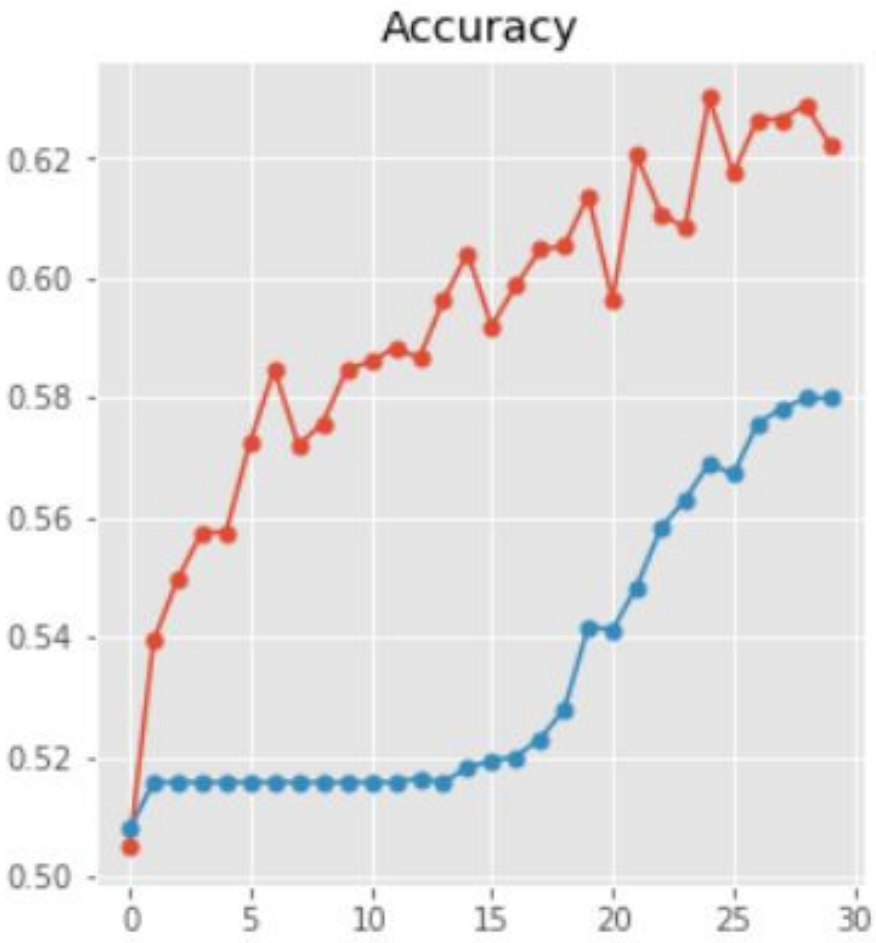
lab filename	lab_tutorial_dnn_practice.ipynb
target	目前是Sigmoid，請嘗試不同的activation function ❑ ReLU、Leaky ReLU ❑ Softplus ❑ Tanh

Try Activation Function

```
+ learning_rate: 0.01
+ loss function: softmax cross entropy
+ optimizer: 基本款 => GradientDescentOptimizer
+ activation function: sigmoid or softplus
```

ReLU vs Sigmoid

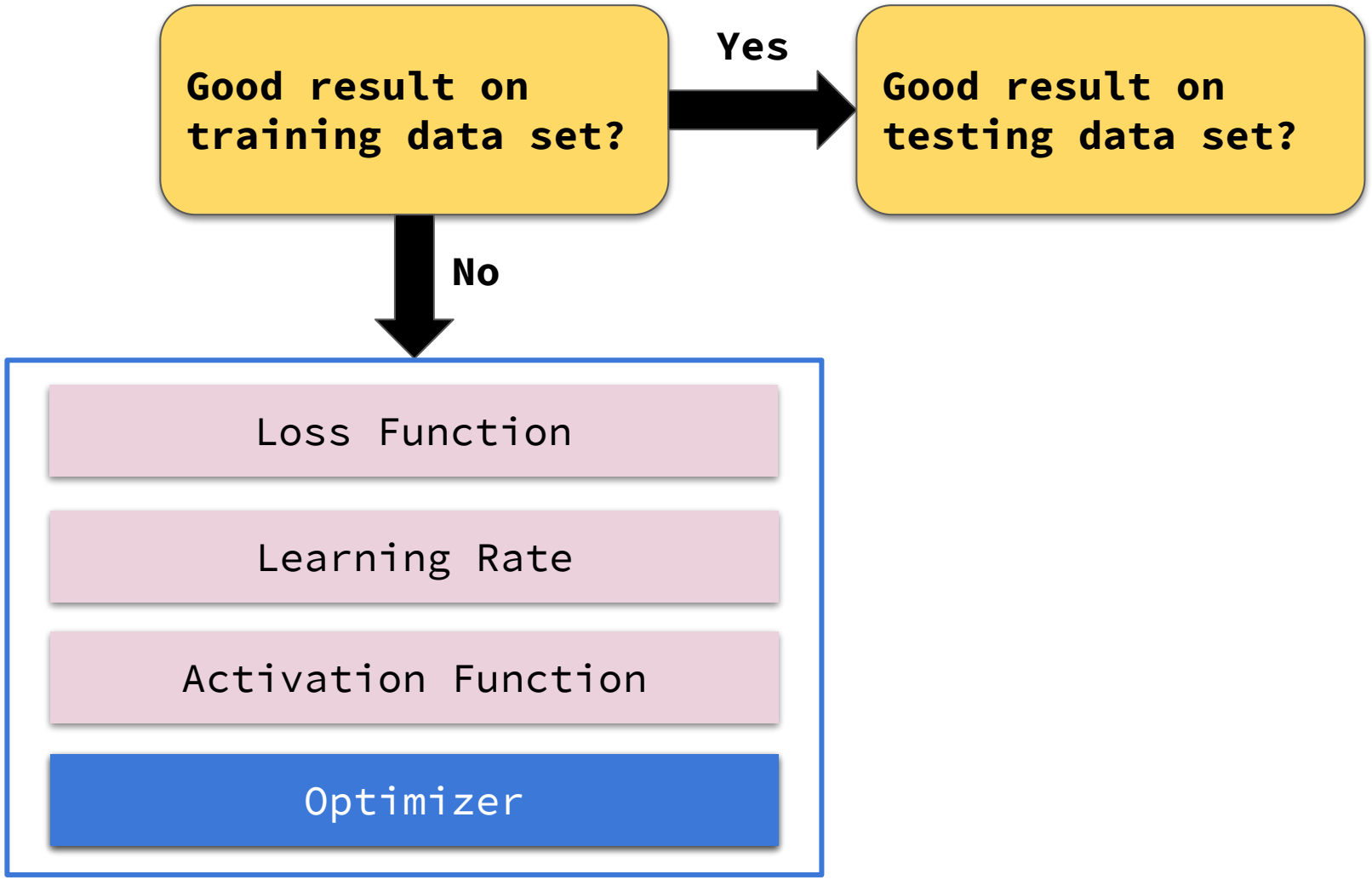




How to Select Activation Function

- Hidden layers
  - 通常會用ReLU
  - Sigmoid有vanishing gradient的問題比較不推薦
- Output layer
  - Regression: linear
  - Classification: softmax or sigmoid

Choose Optimizer



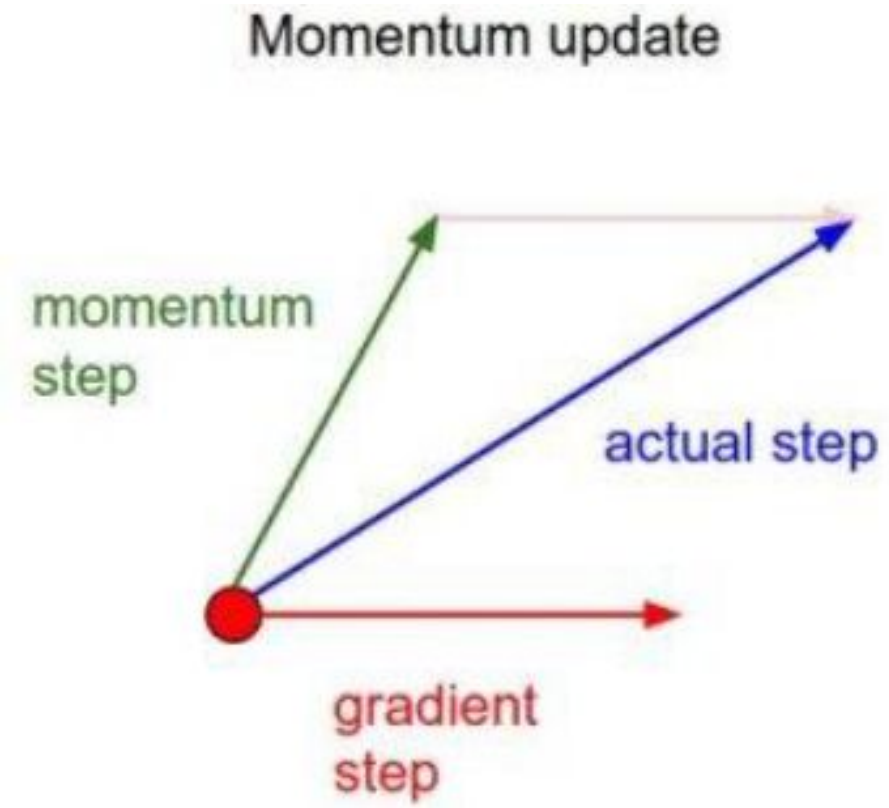
Deep Neural Network(DNN) Tips

Optimizer

- ❏ Gradient Descent
- ❏ Momentum
- ❏ Adagrad - Adaptive Learning Rate
- ❏ RMSprop - Similar with Adagrad
- ❏ Adam - Similar with RMSprop + Momentum

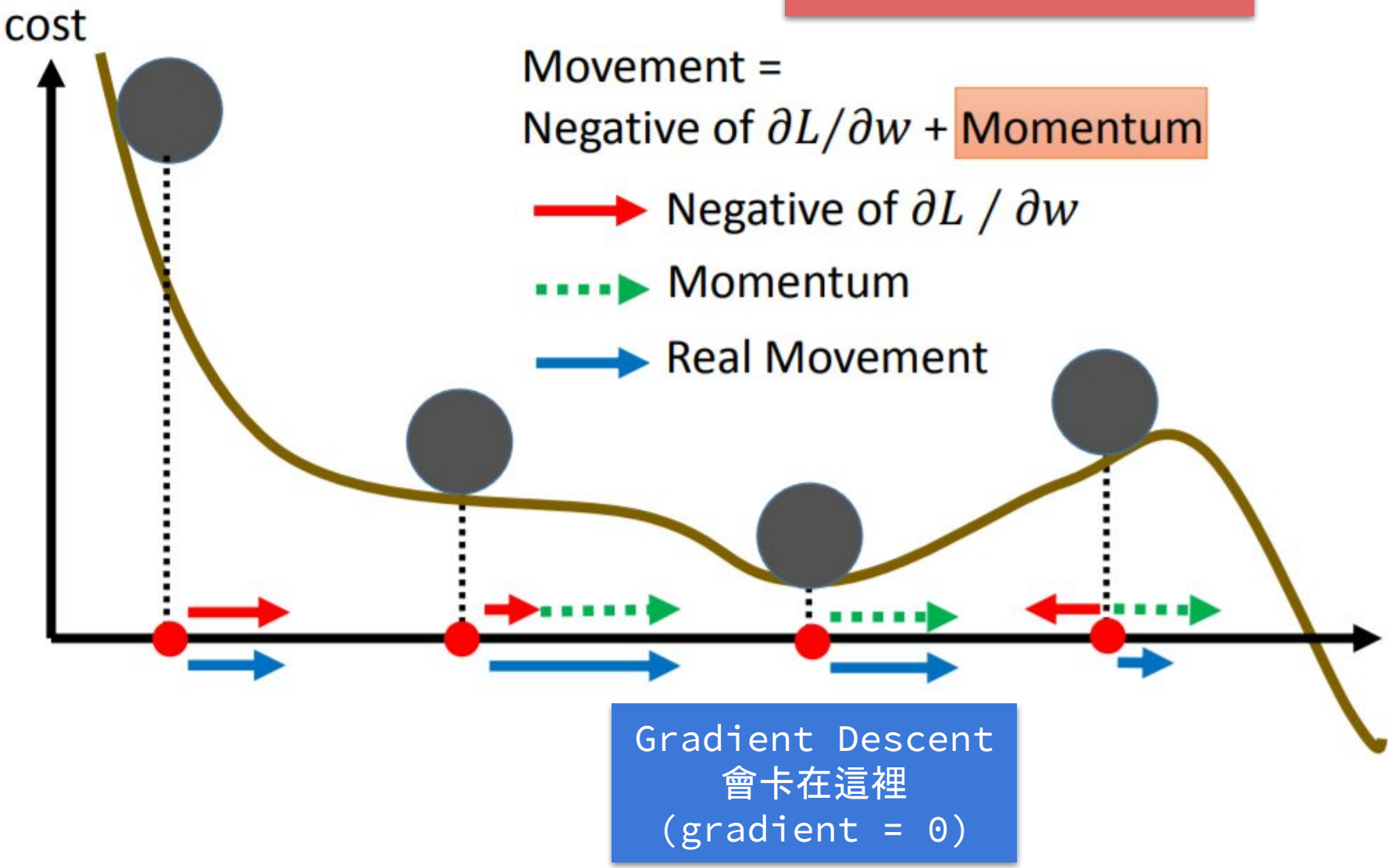
Optimizer – Momentum

- 先算gradient
- 再算momentum
- gradient + momentum ⇒ 更新



Optimizer – Momentum

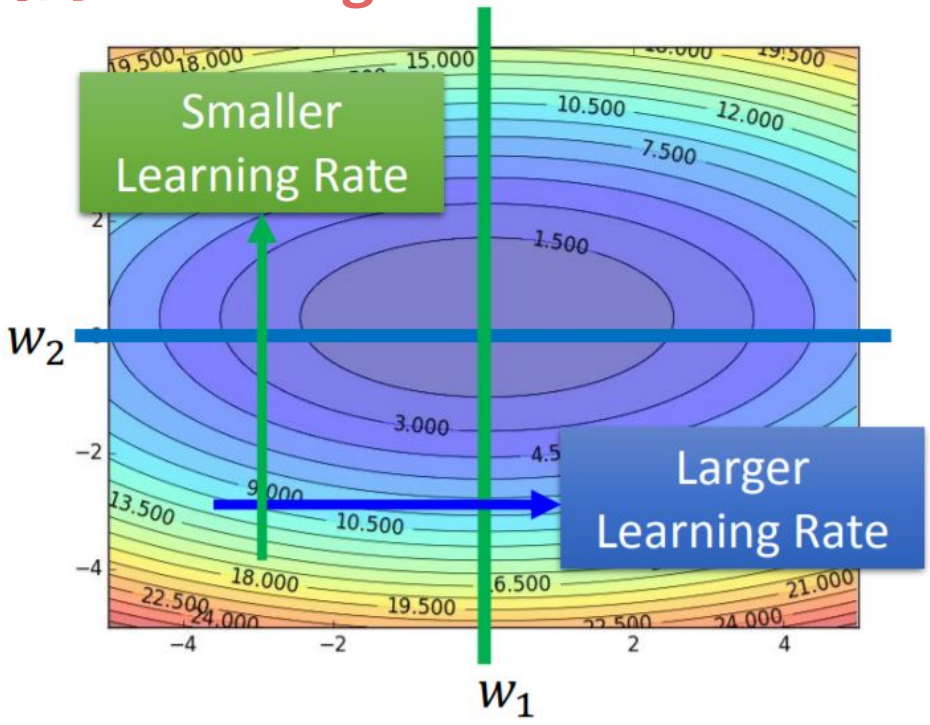
當然沒辦法保證會跑到 global minimum



Optimizer – Adaptive Learning Rate

- 認為一開始離目的很遠，所以用比較大的learning rate
- 在train好幾個epoch之後，離目的越來越近，需要較小的learning rate
- learning rate需要在training runtime調整

- **因材施教：每個參數都有不同的learning rate**



Optimizer - Adagrad

Vanilla Gradient descent

$$\eta^t = \frac{\eta}{\sqrt{t+1}} \qquad g^t = \frac{\partial L(\theta^t)}{\partial w}$$

$$w^{t+1} \leftarrow w^t - \eta^t g^t$$

w is one parameters

Adagrad

$$w^{t+1} \leftarrow w^t - \frac{\eta^t}{\sigma^t} g^t$$

σ^t: ***root mean square*** of the previous derivatives of parameter w

Parameter dependent

Optimizer - Adagrad

- Step by Step

$$\eta^t = \frac{\eta}{\sqrt{t+1}} \qquad g^t = \frac{\partial L(\theta^t)}{\partial w}$$

$$w^1 \leftarrow w^0 - \frac{\eta^0}{\sigma^0} g^0 \qquad \sigma^0 = \sqrt{(g^0)^2}$$

$$w^2 \leftarrow w^1 - \frac{\eta^1}{\sigma^1} g^1 \qquad \sigma^1 = \sqrt{\frac{1}{2}[(g^0)^2 + (g^1)^2]}$$

$$w^3 \leftarrow w^2 - \frac{\eta^2}{\sigma^2} g^2 \qquad \sigma^2 = \sqrt{\frac{1}{3}[(g^0)^2 + (g^1)^2 + (g^2)^2]}$$

$$\vdots$$

$$w^{t+1} \leftarrow w^t - \frac{\eta^t}{\sigma^t} g^t \qquad \sigma^t = \sqrt{\frac{1}{t+1}\sum_{i=0}^{t}(g^i)^2}$$

Optimizer – Adagrad

$$w^{t+1} \leftarrow w^t - \frac{\eta^t}{\sigma^t} g^t$$

$$\eta^t = \frac{\eta}{\sqrt{t+1}}$$ 1/t decay

$$\sigma^t = \sqrt{\frac{1}{t+1}\sum_{i=0}^{t}(g^i)^2}$$

$$w^{t+1} \leftarrow w^t - \frac{\eta}{\sqrt{\sum_{i=0}^{t}(g^i)^2}} g^t$$

一次微分

二次微分（近似）

Optimizer – Adagrad

- 直覺的理解

特別大

g^0	g^1	g^2	g^3	g^4	...
0.001	0.001	0.0002	0.0087	0.1	...

g^0	g^1	g^2	g^3	g^4	...
8.7	10.87	18.7	87.7	0.1	...

特別小

$$w^{t+1} \leftarrow w^t - \frac{\eta}{\sqrt{\sum_{i=0}^{t}(g^i)^2}} g^t$$

- 老馬識途：參考之前的經驗修正現在的步伐
- 不完全相信當下的gradient
- 缺點：learning rate會一直遞減，越train越慢...

Lab: gradient_demo.ipynb (5 minutes)

lab filename	gradient_demo.ipynb
target	Gradient descent與Adagrad的簡單比較

Optimizer – RMSProp

Adagrad

$$w^{t+1} = w^t - \frac{\eta}{\sigma^t} g^t$$

$$\sigma^t = \sqrt{(g^0)^2 + \ldots + (g^t)^2}$$

RMSProp

$$w^{t+1} = w^t - \frac{\eta}{\sigma^t} g^t$$

$$\sigma^t = \sqrt{\alpha(\sigma^{t-1})^2 + (1-\alpha)(g^t)^2}$$

α小：代表相信當前gradient

α大：代表相信之前的gradient

- **另一種參考過去gradient的方式**

Optimizer - Adam(Adaptive Moment Estimation)

- Close to RMSProp + Momentum

$$m^t = \beta_1 m^t + (1-\beta_1) g^t$$

$$\sigma^t = \beta_2 \sigma^t + (1-\beta_2)(g^t)^2$$

$$\eta^t = \eta^t \frac{\sqrt{1-\beta_2^t}}{(1-\beta_1^t)}$$

$$w^{t+1} = w^t - \eta \frac{m^t}{\sqrt{\sigma^t + \epsilon}} g^t$$

For Momentum

For RMSProp

β_1: m的衰減係數

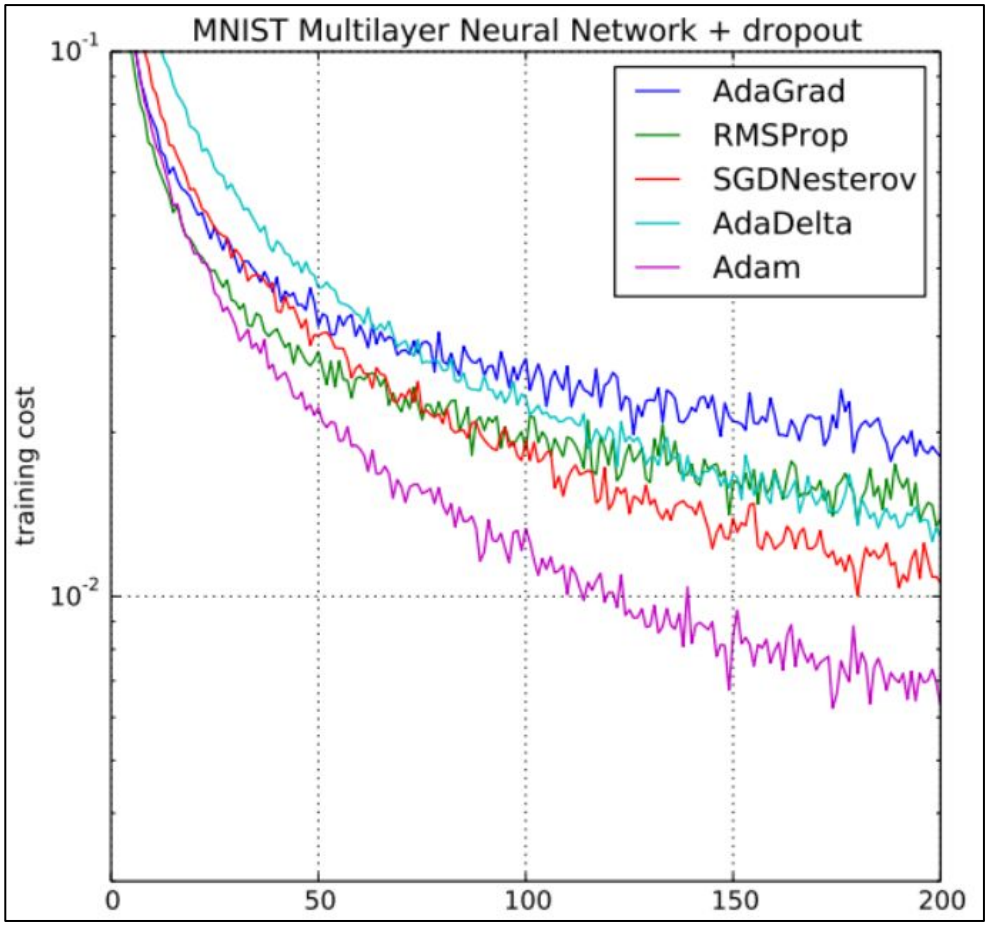
β_2: σ的衰減係數

```
tf.train.AdamOptimizer(learning_rate=0.001, beta1=0.9, beta2=0.999, epsilon=1e-08)
```

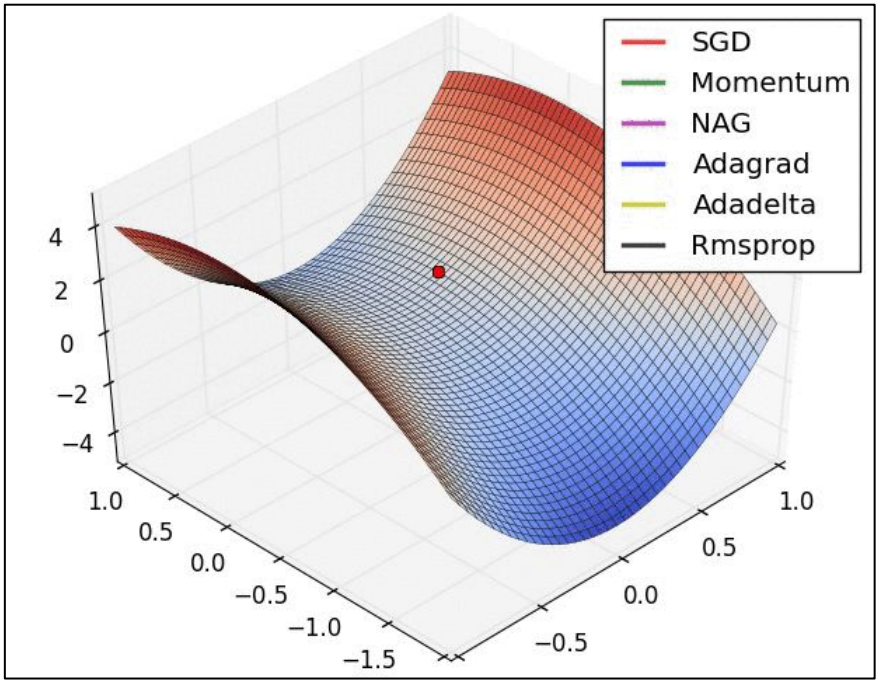
Paper: Adaptive Moment Estimation

實際上不用調整參數已經有很好的效果

比較Optimizers



- 一般起手式：Adam
- 實際上Optimizer的效果還是會跟資料有關，沒有絕對要用什麼工具的規則！



Lab: lab_dnn_tutorial_practice.ipynb (10 ~ 15 minutes)

lab filename	lab_tutorial_dnn_practice.ipynb
target	使用不同的Optimizer: ❑ tf.train.AdamOptimizer ❑ tf.train.AdagradOptimizer ❑ tf.train.MomentumOptimizer ❑ tf.train.RMSPropOptimizer

Try Different Optimizer

+ learning_rate: 0.01
+ loss function: softmax cross entropy
+ optimizer: GradientDescentOptimizer or AdamOptimizer
+ activation function: softplus

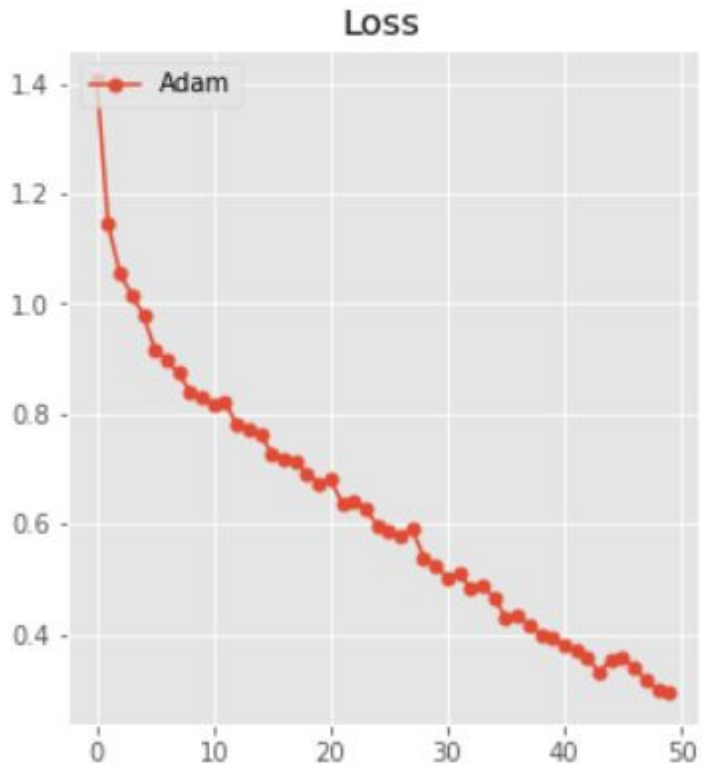
Result: Gradient Descent vs Adam

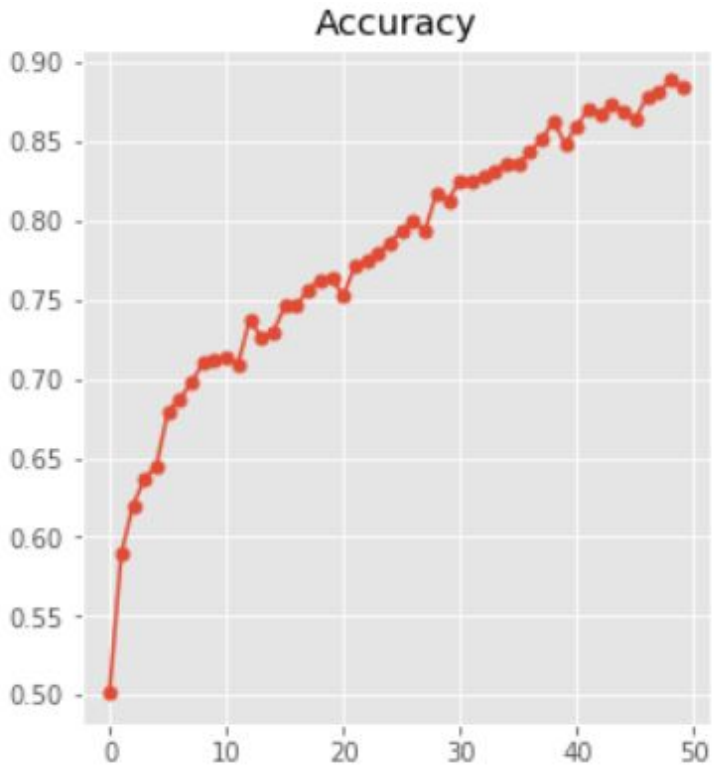
```
model with Adam optimizer!
30/30.  train loss: 0.773, valid loss: 0.975, train acc: 0.732, valid acc: 0.696

model with SGD optimizer!
30/30.  train loss: 1.095, valid loss: 1.434, train acc: 0.605, valid acc: 0.484

<matplotlib.text.Text at 0x7f2ca857d7f0>
```

Current Best Model Configuration

Loss function	**cross entropy**
Activation function	**relu + softmax**
Optimizer	**Adam**

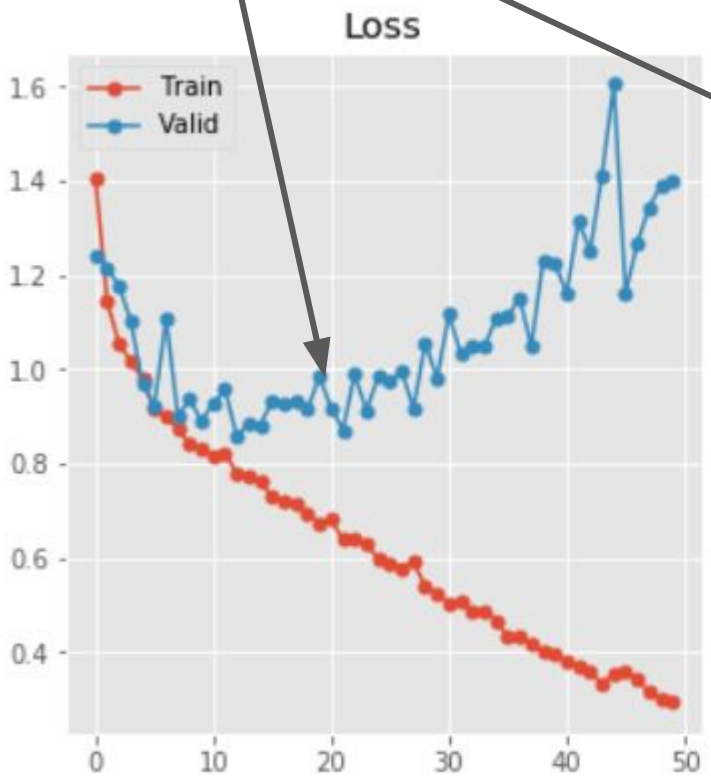
50 epochs後準確率達90%, 可以舉杯慶祝了!!!

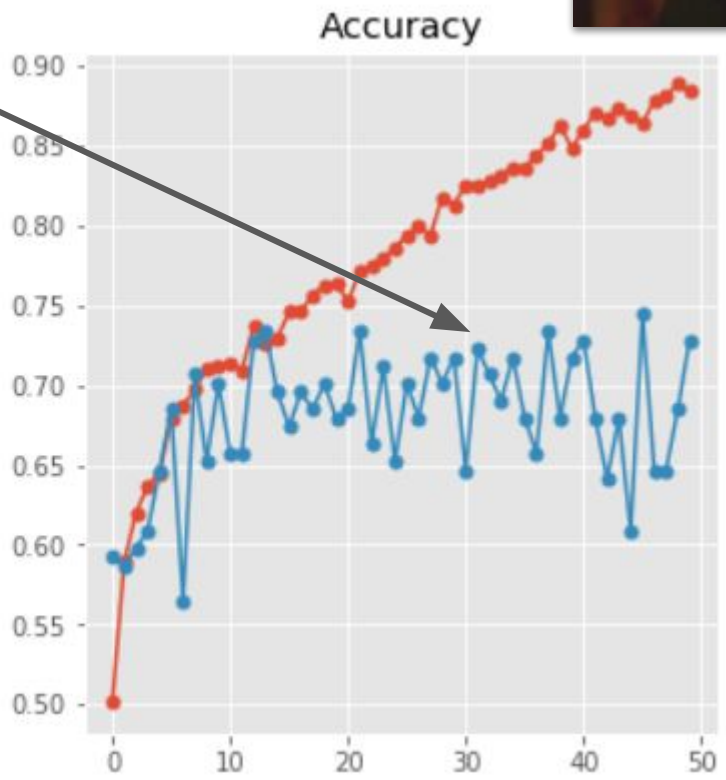




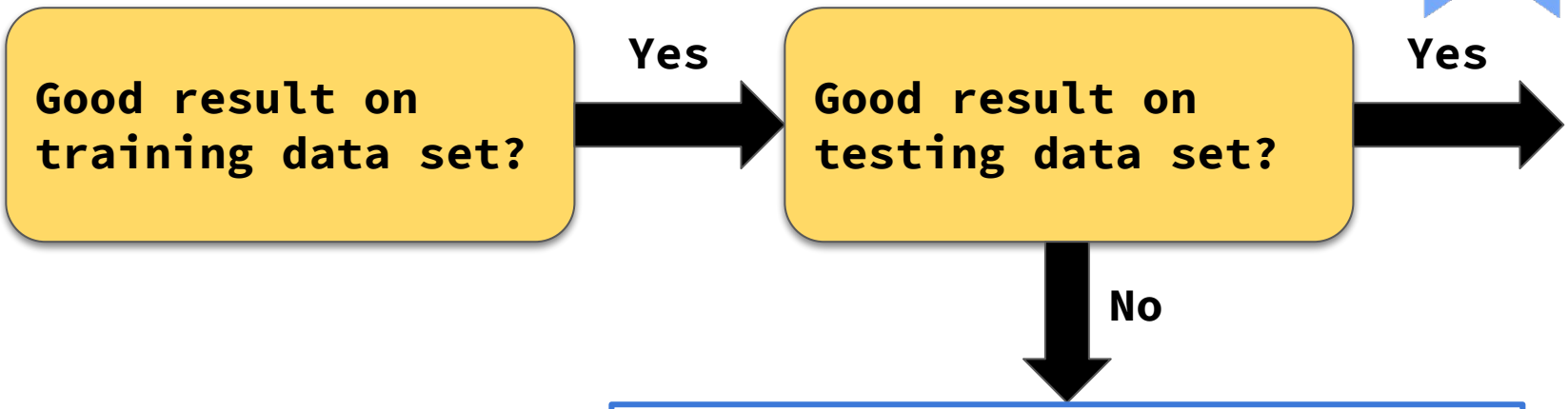
果然...世事難預料!

Overfitting!





Overfitting



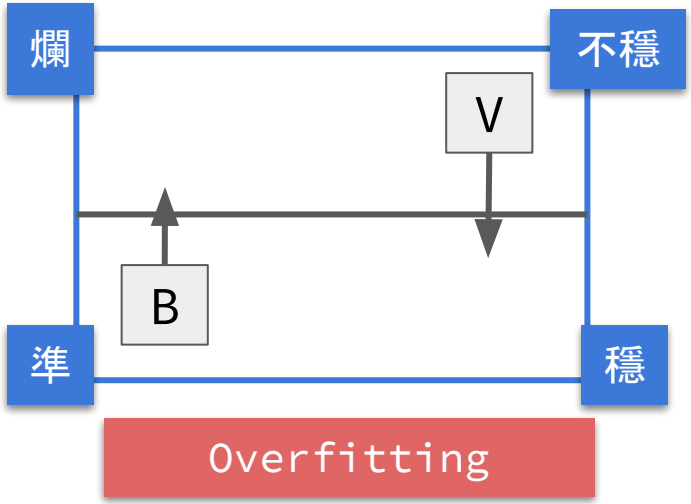
Overfitting的意義:

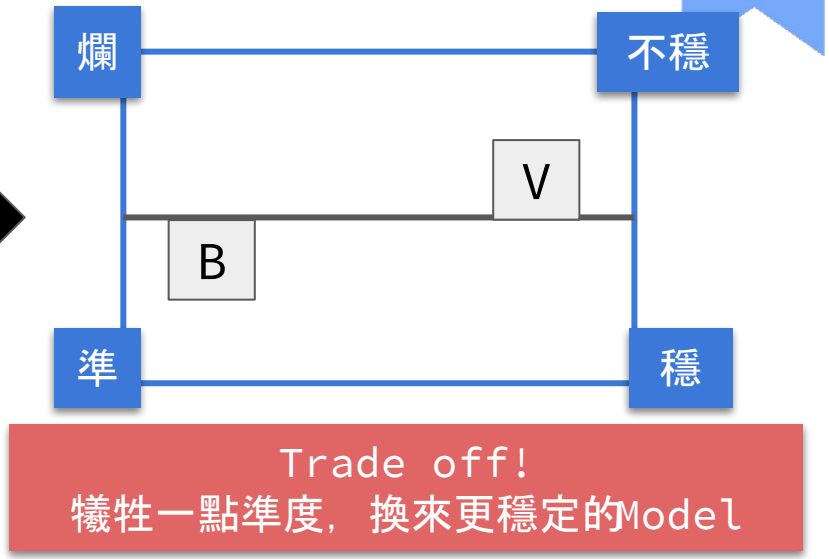
代表model已經太過於擬合data，必須限制model的能力，**達到generalization的目的**

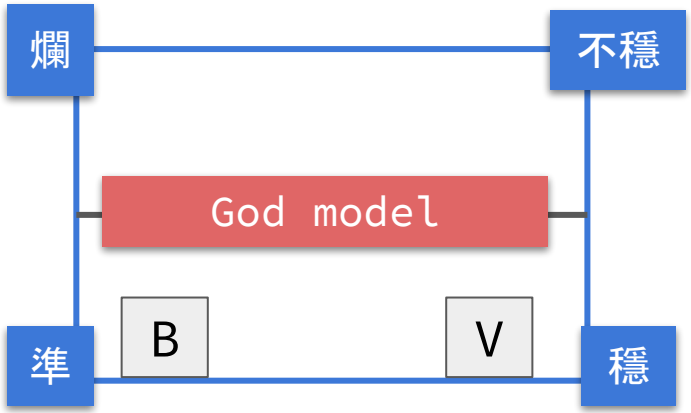
- Early Stopping
- Regularization
- Dropout

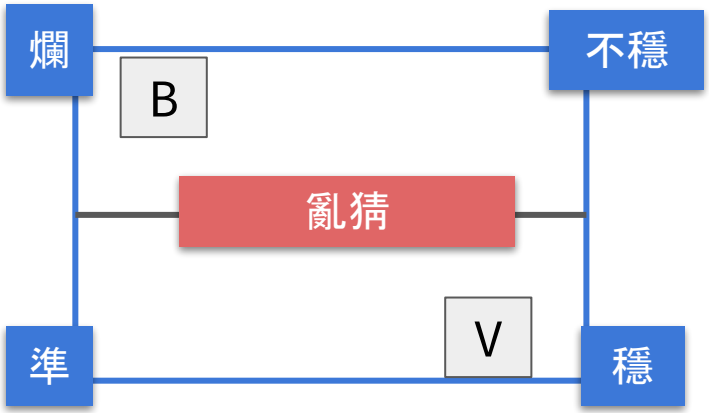
Deep Neural Network(DNN) Tips

Model Bias and Variance

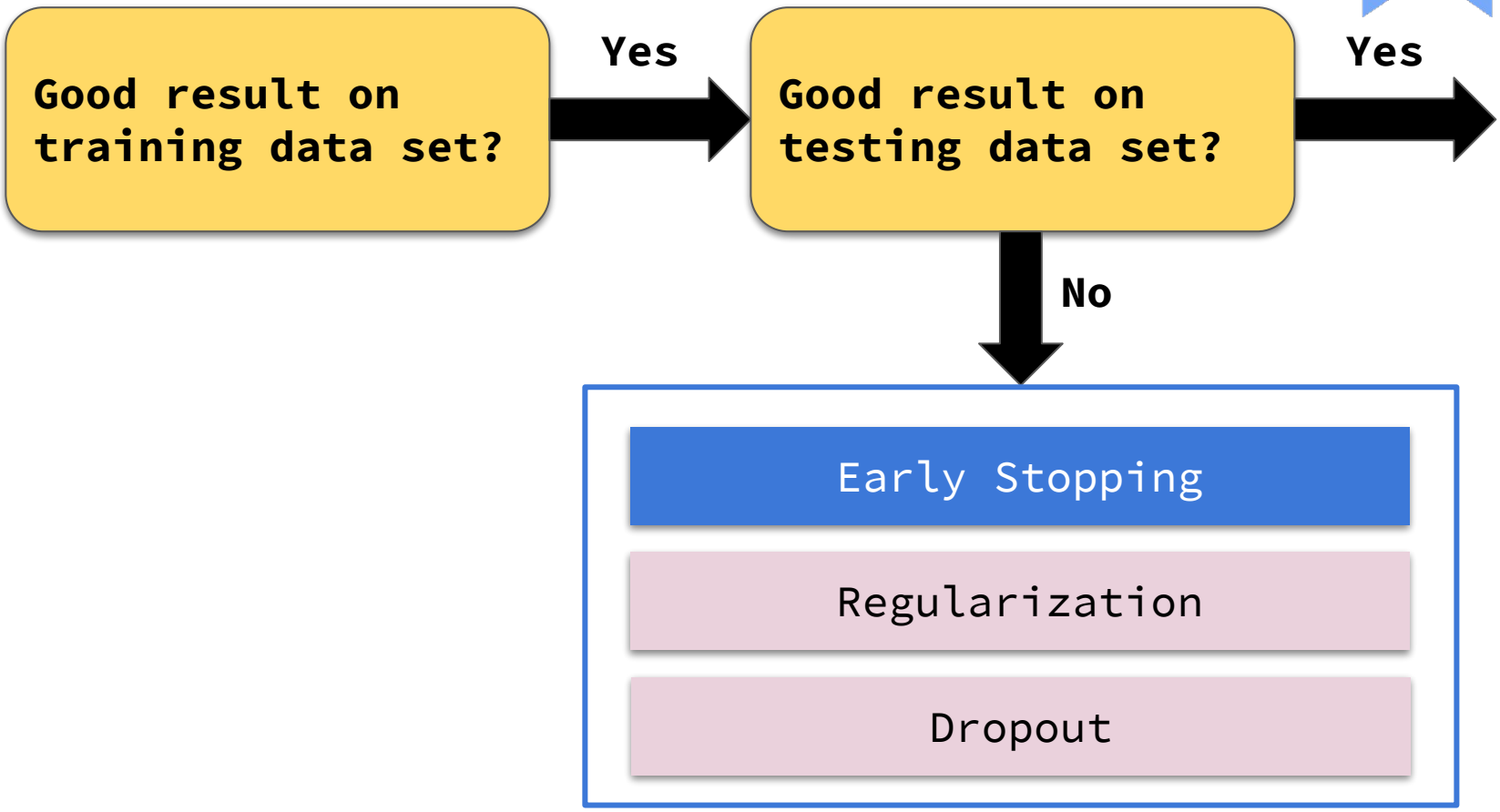






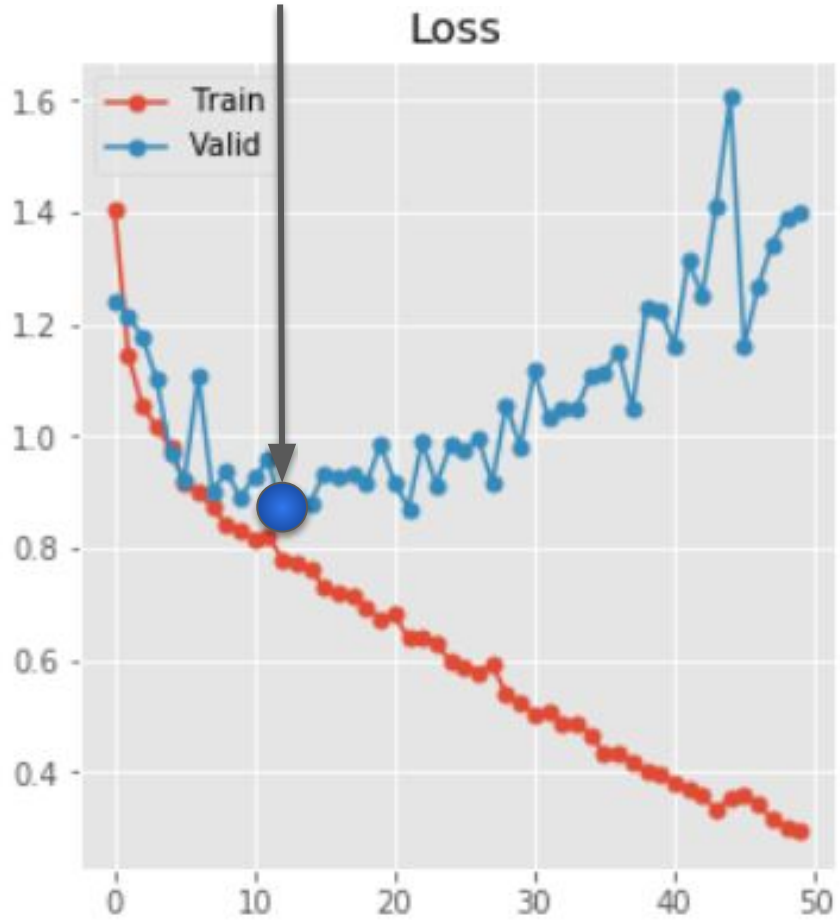


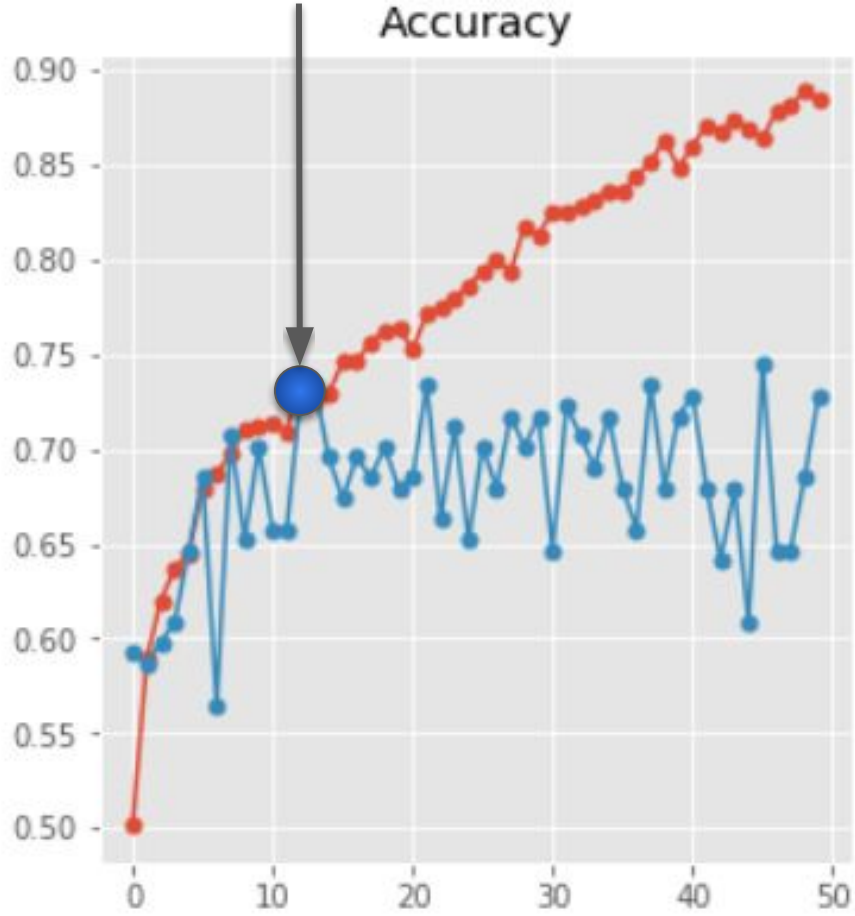
Overfitting



Early Stopping: 懸崖勒馬!

早一點停下來就沒事了!





Early Stopping: 懸崖勒馬!

```
tf.reset_default_graph()
model_adam = ModelAdam()
model_adam.fit(data_fn(X_train, Y_train, n_batch, shuffle=True),
               data_fn(X_valid, Y_valid, n_batch, shuffle=False),
               n_epoch,
               lr=learning_rate,
               callback=EarlyStopping(thres=5))
```

Model fit加上這個callback object即可!

Early Stopping: 懸崖勒馬!

fit 有這麼一段, check callback function on epoch end

```
for ep in range(1, n_epoch + 1):
    training …

    if callback is not None:
        callback(self)
```

```
def __call__(self, model):
    last_vl_loss = model.hist["vl_loss"][-1]
    if self.min_loss > last_vl_loss:
        self.patient = 0
        self.min_loss = last_vl_loss
    else:
        self.patient += 1
    if self.patient >= self.thres:
        raise StopIteration("\nEarly Stopping, valid loss keep increasing for {} times ! "
                            "lowest loss is {:.3f} !"
                            .format(self.patient, self.min_loss))
```

EarlyStopping物件

validation loss是否持續增加?
超過一定threshold則停止

Lab: lab_dnn_tutorial_practice.ipynb (5 ~ 10 minutes)

lab filename	lab_tutorial_dnn_practice.ipynb
target	加入callback object: EarlyStopping check if overfitting

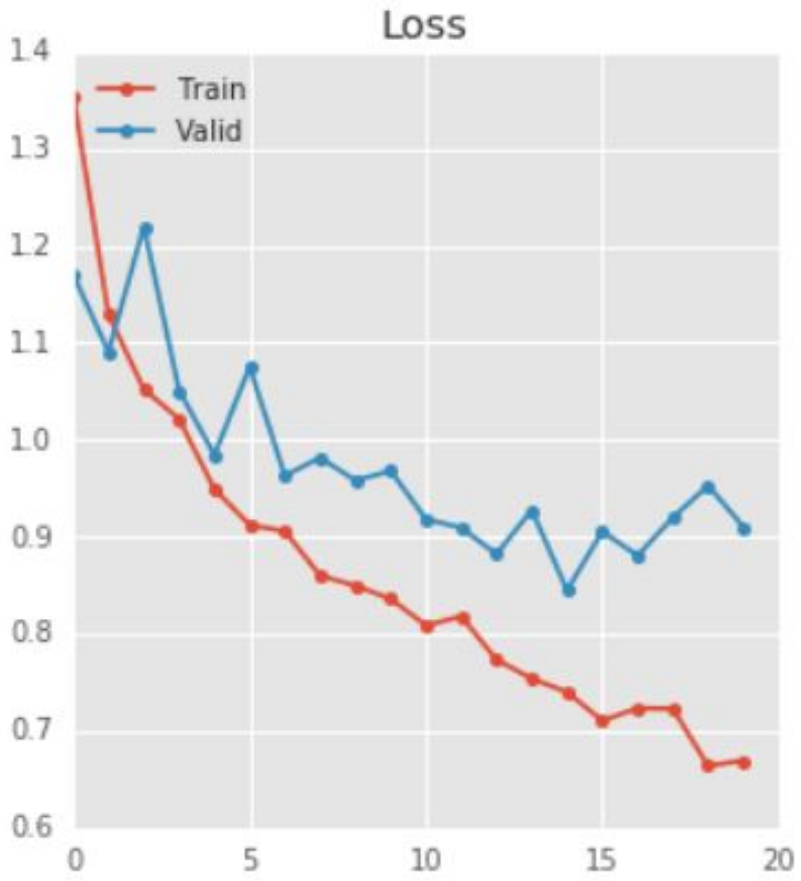
Solve Overfitting: EarlyStopping

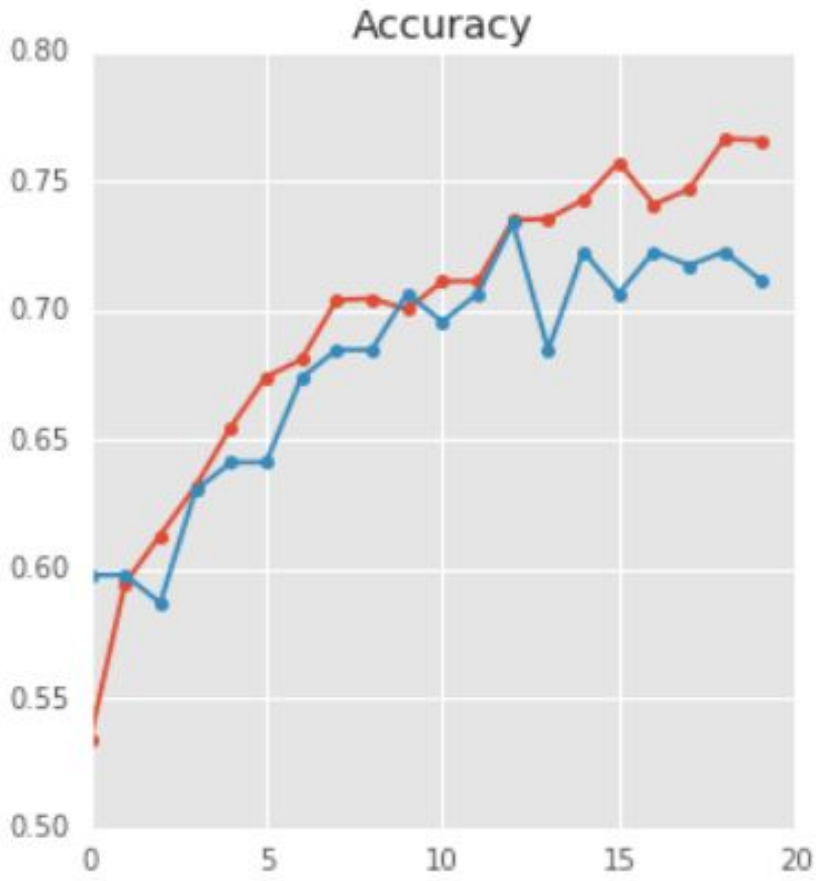
檢查Valid Loss變化, 若發現Loss持續增加超過一定Threshold, 則停止Training

+ learning_rate: 0.001
+ loss function: softmax cross entropy
+ optimizer: AdamOptimizer
+ activation function: softplus
+ number of train epoch: 50

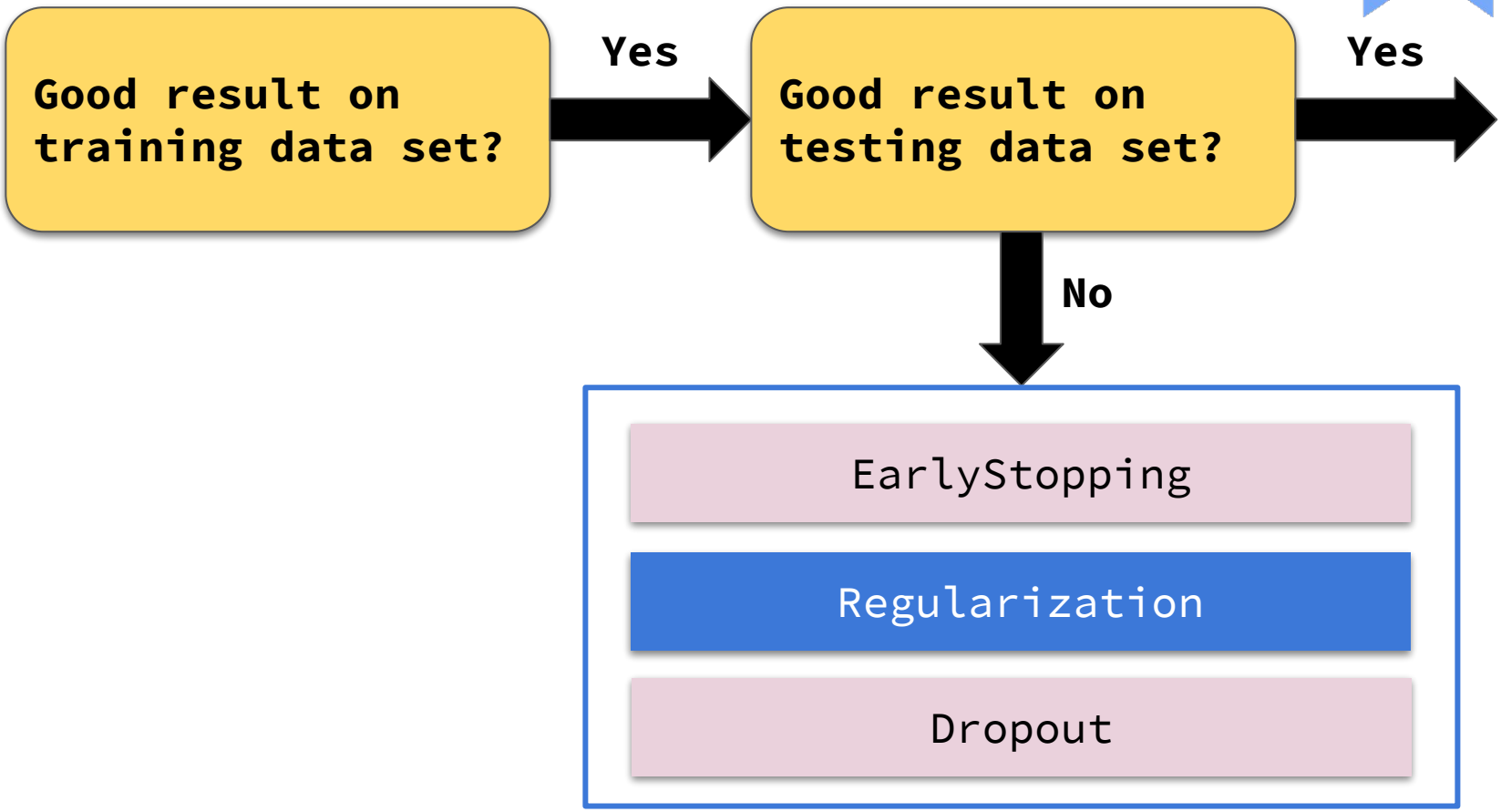
Early Stopping - Threshold=5

```
19/50.  train loss: 0.664, valid loss: 0.953, train acc: 0.766, valid acc: 0.723
Early Stopping, valid loss keep increasing for 5 times ! lowest loss is 0.845 !
```



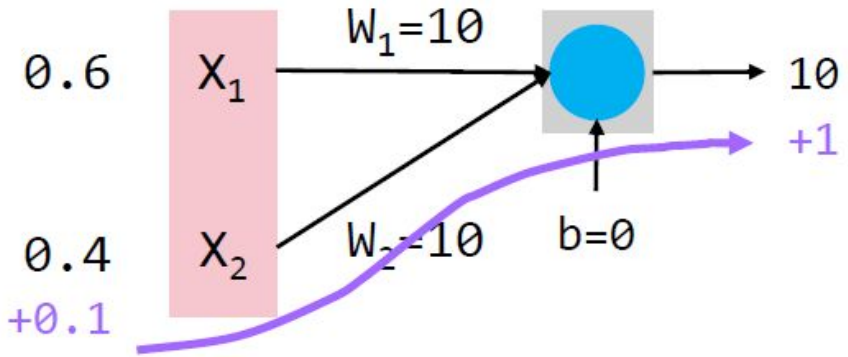


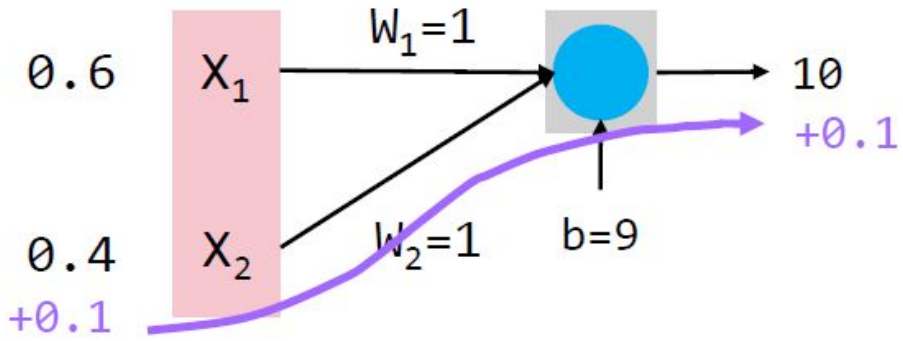
Overfitting



Regularization

- Loss限制weights 的大小讓output 曲線比較平滑





w_i 較小 ➔ Δx_i 對 $\hat{y}$ 造成的影響($\Delta\hat{y}$)較小
➔ 對 input 變化比較不敏感 ➔ generalization 好

Regularization

- 作法：加入目標(Loss)函數中，一起優化

$$L'(\theta) = L(\theta) + \lambda(regularizer)$$

- λ是用來調整regularization的比重(通常在0.01以下)
- Regularizer兩種形式：L1 and L2

$$L_1 = \sum_i |w_i|$$

L1 norm: sum of absolute values

$$L_2 = \sum_i |w_i|^2$$

L2 norm: square sum of absolute values

一般不會把bias納入regularizer term

Regularization: L2 Norm

L2 regularization:

$$\|\theta\|_2 = (w_1)^2 + (w_2)^2 + \dots$$

$$L'(\theta) = L(\theta) + \lambda \frac{1}{2}\|\theta\|_2$$ Gradient: $$\frac{\partial L'}{\partial w} = \frac{\partial L}{\partial w} + \lambda w$$

Update:

$$w^{t+1} \rightarrow w^t - \eta \frac{\partial L'}{\partial w} = w^t - \eta\left(\frac{\partial L}{\partial w} + \lambda w^t\right)$$

$$= (1-\eta\lambda)w^t - \eta \frac{\partial L}{\partial w}$$

接近0

Weight Decay

Lab: lab_dnn_tutorial_practice.ipynb (5 ~ 10 minutes)

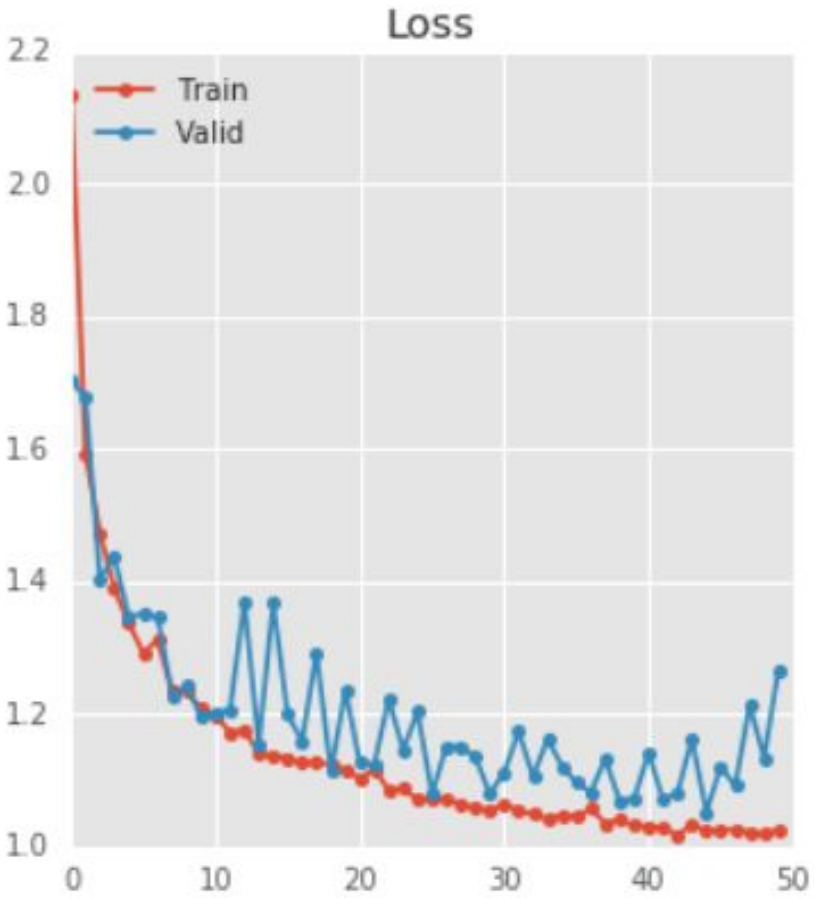
lab filename	lab_tutorial_dnn_practice.ipynb
target	加入l2 regularizer: tf.contrib.layers.l2_regularizer tf.contrib.layers.l1_regularizer

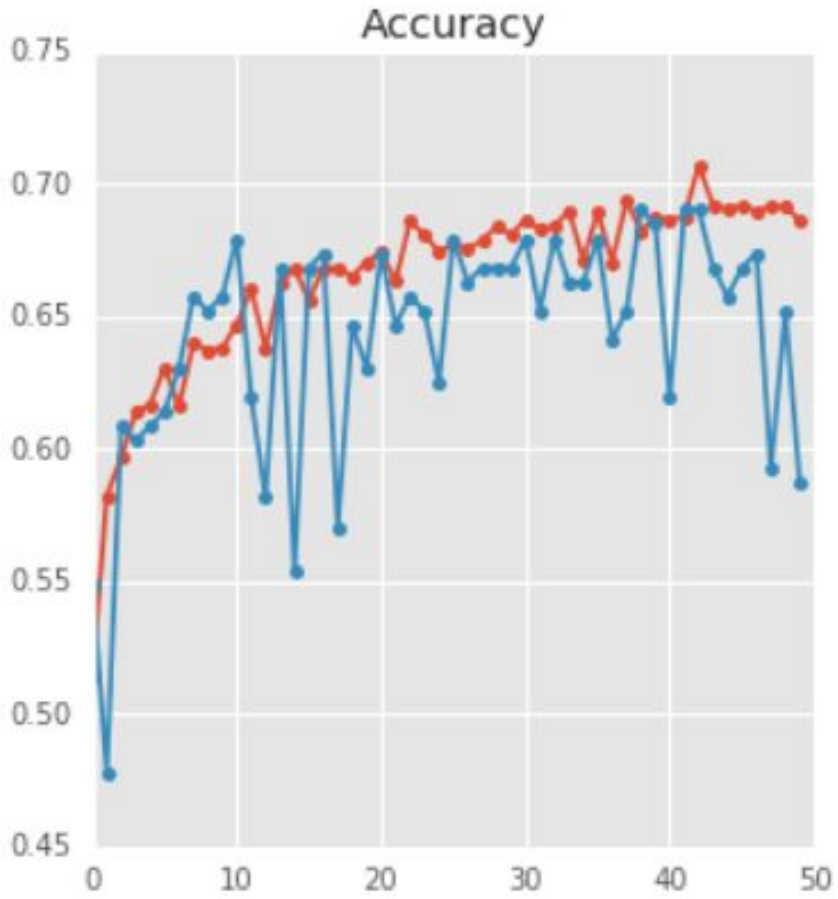
Solve Overfitting: Add Regularizer Term to Loss Function

```
+ learning_rate: 0.001
+ loss function: softmax cross entropy
+ optimizer: AdamOptimizer
+ activation function: softplus
+ number of train epoch: 50

+ >>> add l2 regularizer term to each hidden layers with scale 0.01
```

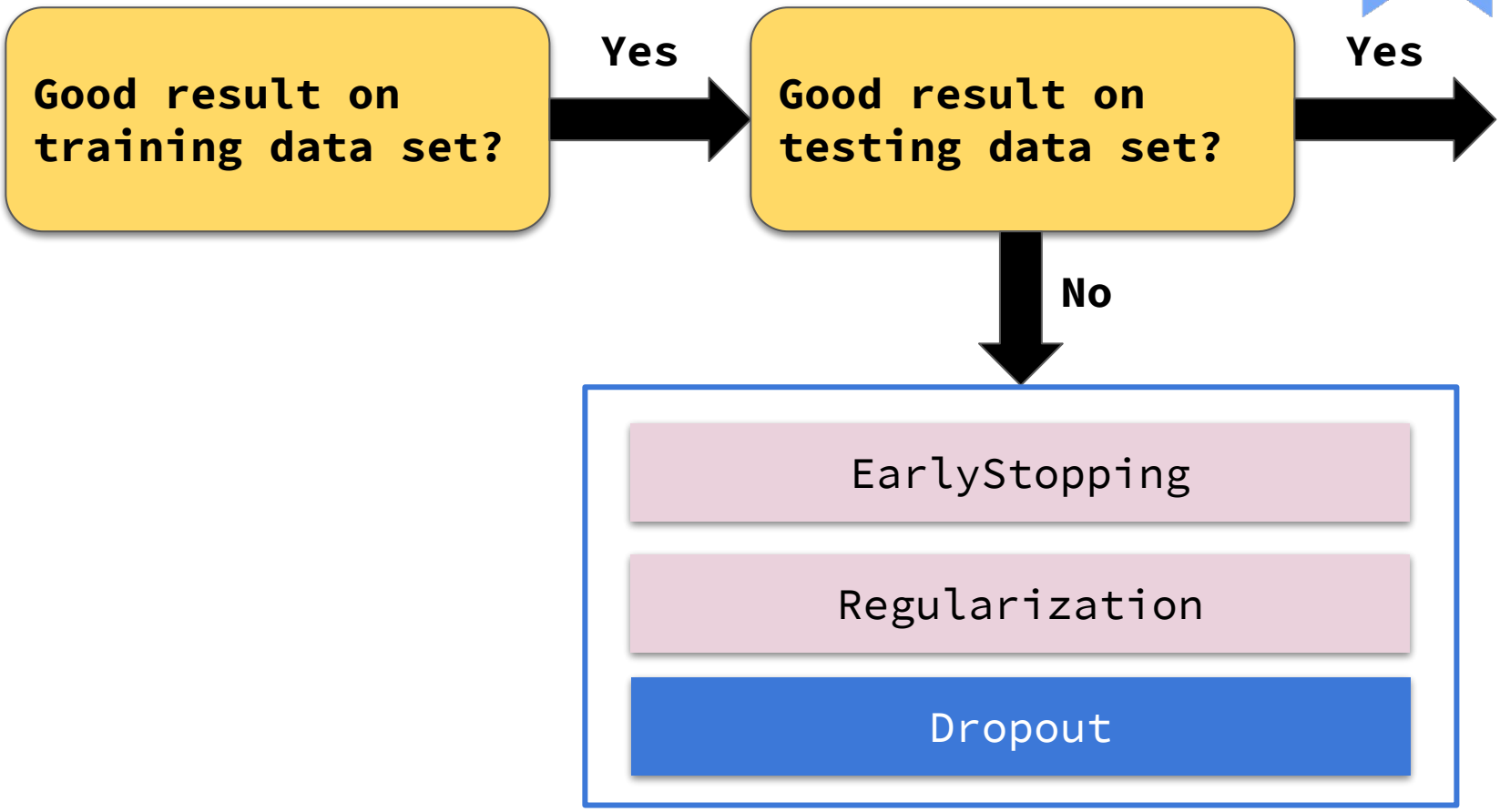
Result - Regularization

```
50/50.  train loss: 1.025, valid loss: 1.268, train acc: 0.686, valid acc: 0.587
```



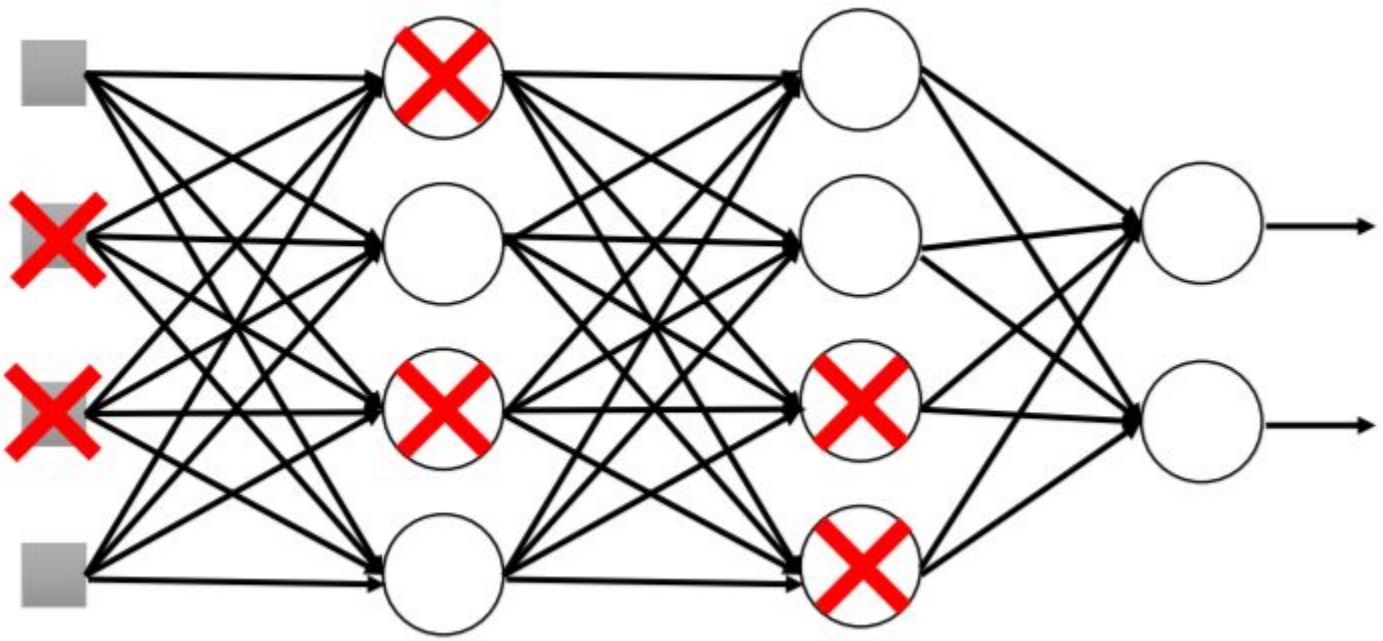


Overfitting



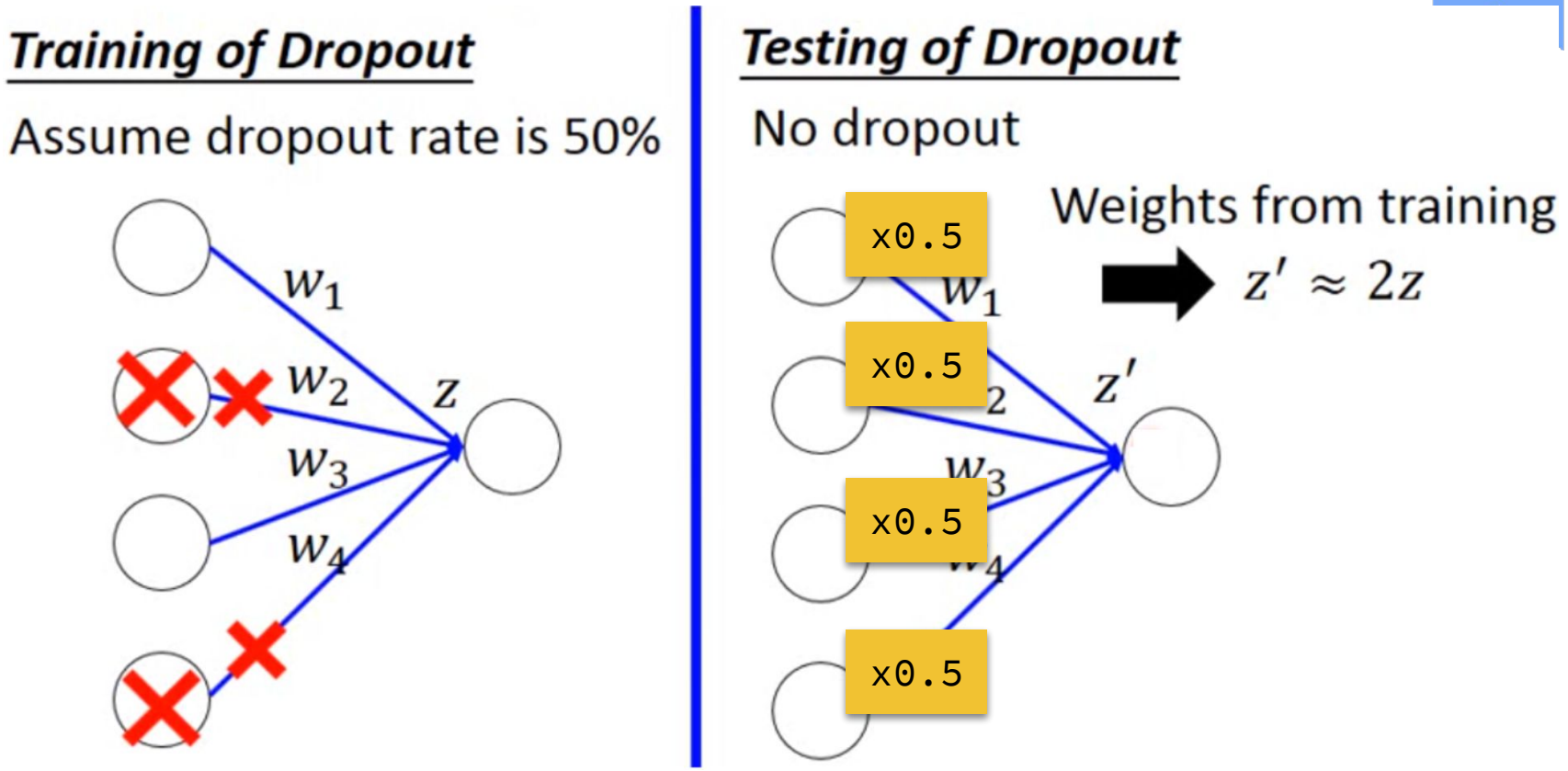
Dropout

Training:



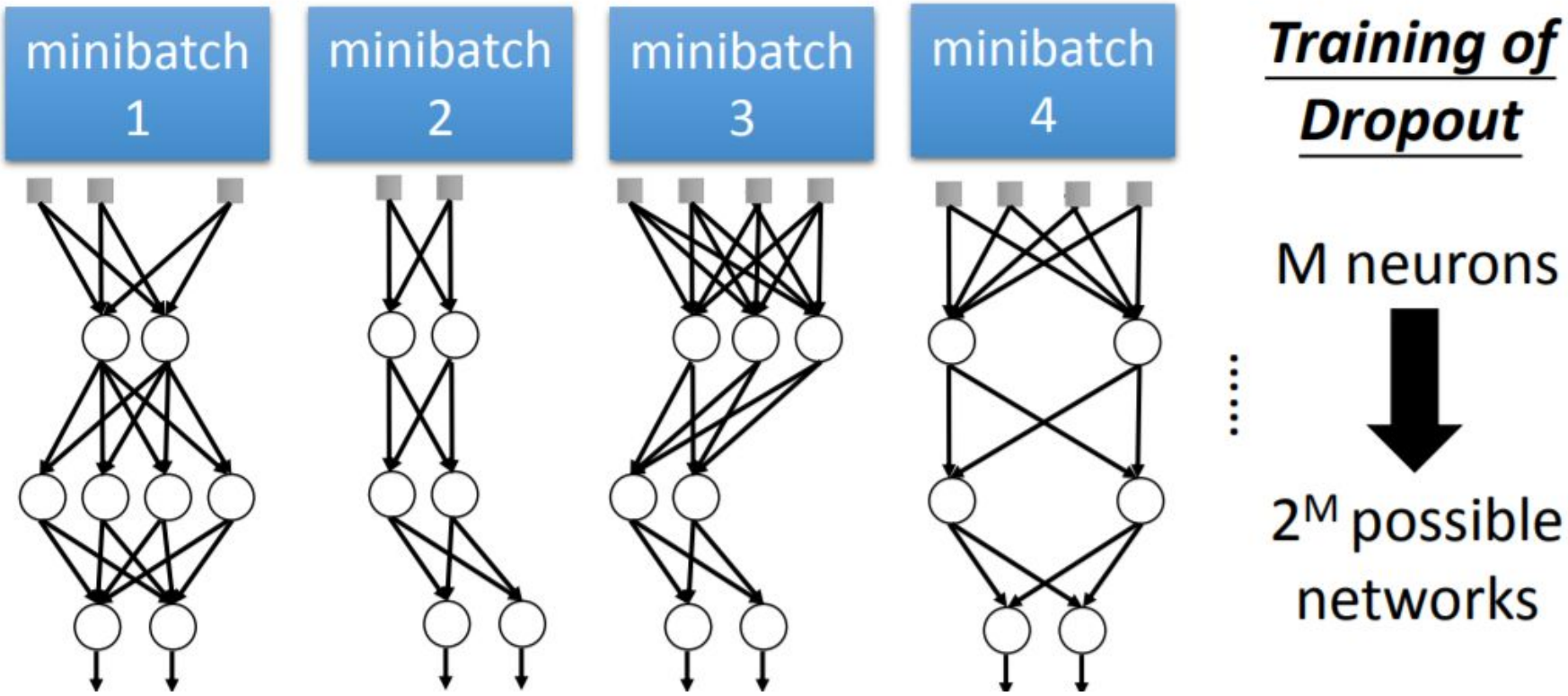
- 每個batch training時都會隨機丟掉p%的neuron (weight設為0 or 乘0))
- **Dropout always work on training time, not on evaluate or prediction**

Dropout: 神妙的地方



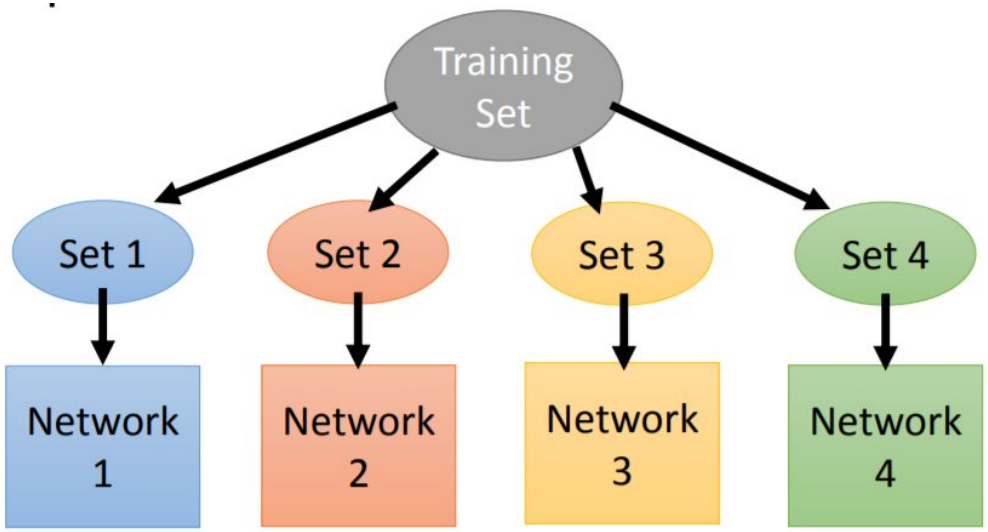
- Testing的時候weight x (1 - p%) ⇒ z'≈ z
- 不用擔心，一般Toolkits都幫你做掉了，除非自己implements

Dropout - 終極的ensemble方法



- Using one mini-batch to train one network
- weight in the network are shared

Dropout - 終極的ensemble方法

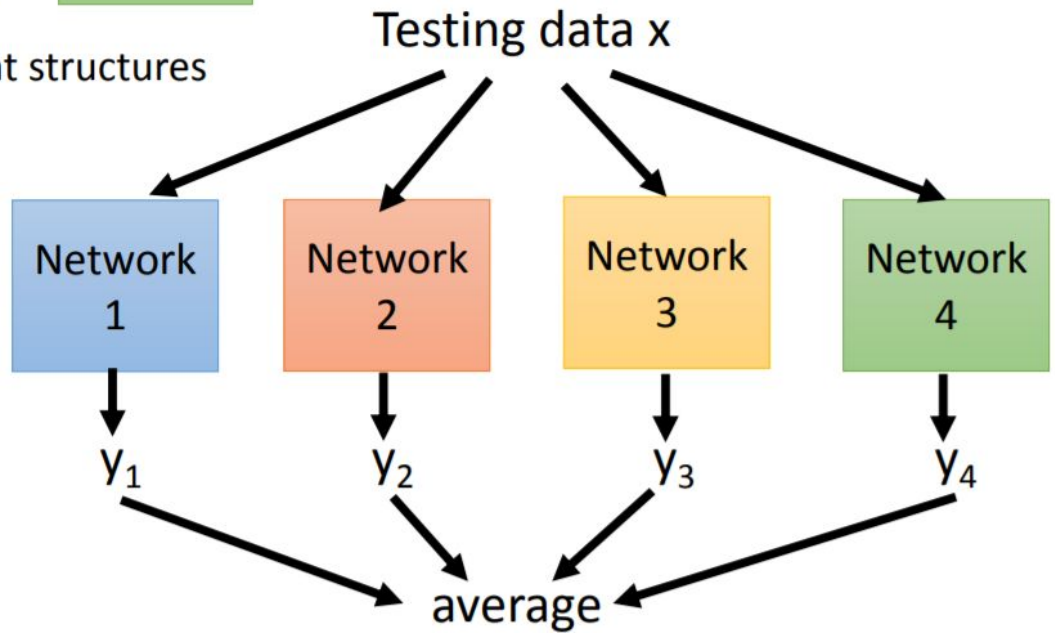


Train a bunch of networks with different structures

Bagging

Decision Tree

Random Forest



直覺的理解

Dropout的結果

- 會讓training performance變差，包含training速度
- **不要一開始就使用dropout!**

Lab: lab_dnn_tutorial_practice.ipynb (5 ~ 10 minutes)

lab filename	**lab_tutorial_dnn_practice.ipynb**
target	**Network加入dropout:** **drop_rate可嘗試 0.3, 0.4, 0.5, 0.6**

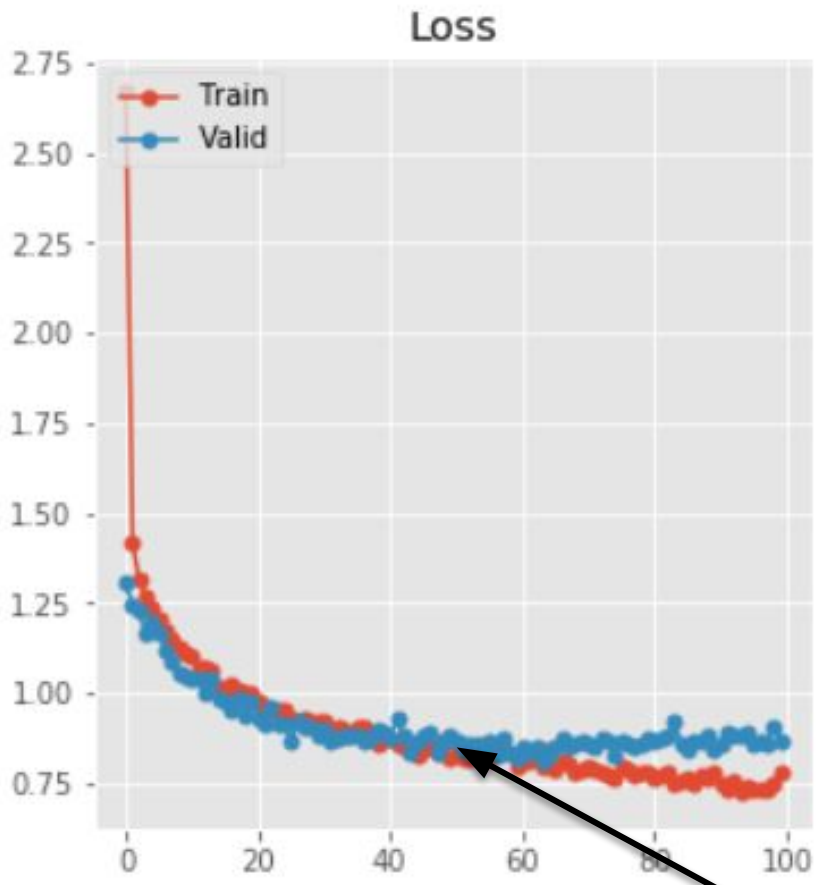
Solve Overfitting: Dropout

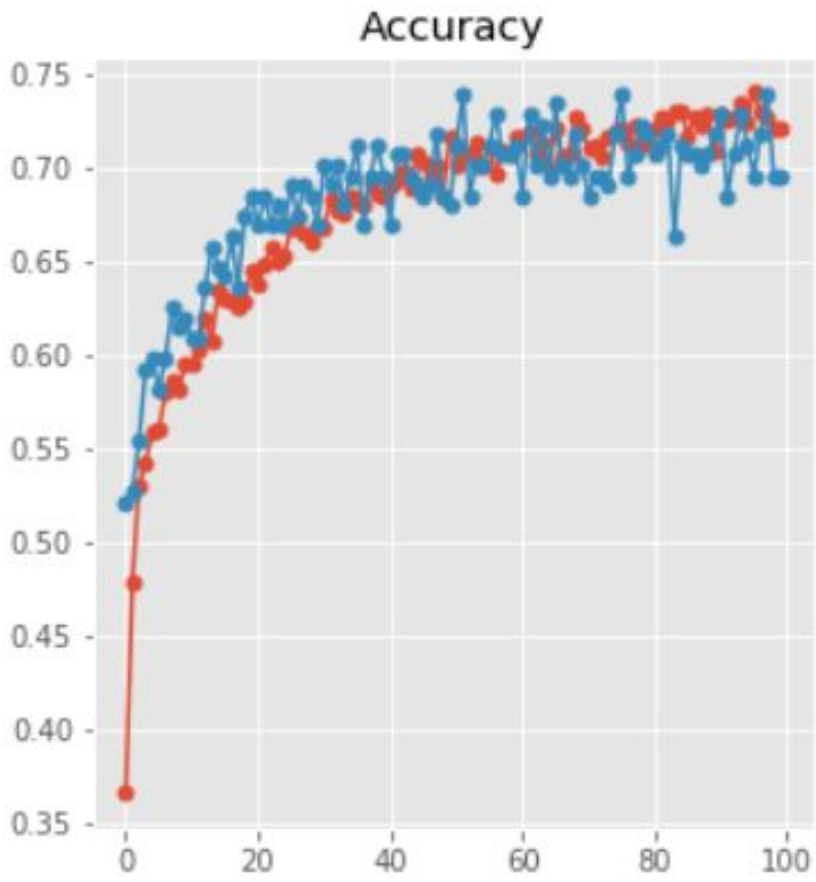
+ learning_rate: 0.001
+ loss function: softmax cross entropy
+ optimizer: AdamOptimizer
+ activation function: relu

+ number of train epoch: 100

>>> Dropout會讓training速度變慢, 增加epoch次數讓分數更好!

Result - Dropout(After 100 Epochs)

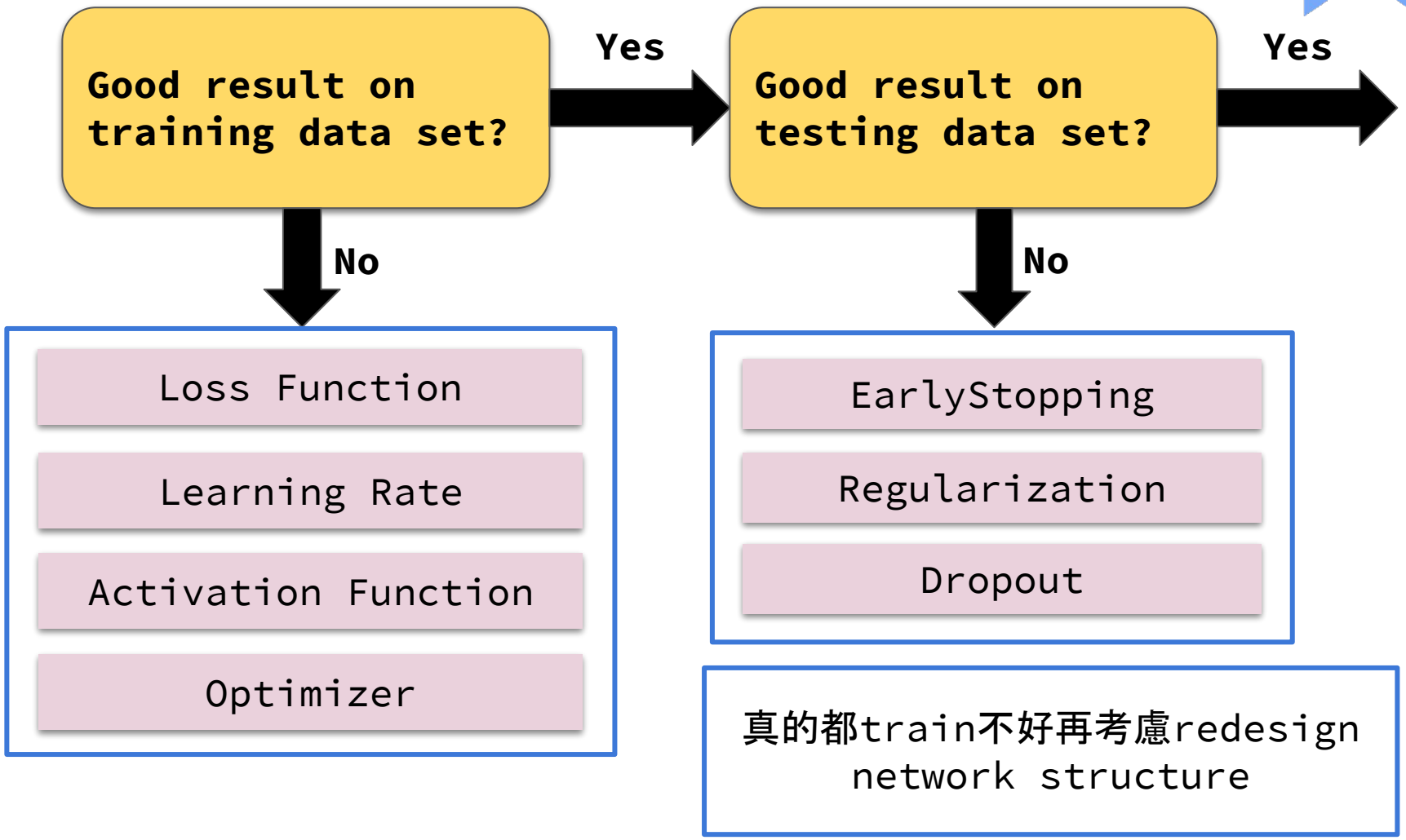




Valid Loss曲線在50 epochs之後已經下降變慢，但並沒有遽增

Deep Neural Network(DNN) Tips

Recall



Recommendation Engine in Matrix Factorization

- Machine Learning Framework
- TensorFlow Fundamentals
- Deep Neural Network(DNN) Tips
- Recommedation Engine in Matrix Factorization
- Matrix Factorization with DNN

Recommendation Engine in Matrix Factorization

Movielens Data

User Movie Rating

	userId	movieId	rating	timestamp
0	0	30	2.5	1260759144
1	0	833	3.0	1260759179
2	0	859	3.0	1260759182
3	0	906	2.0	1260759185
4	0	931	4.0	1260759205

User Movie Tag

	userId	movieId	tag	timestamp
0	14	304	sandra 'boring' bullock	1138537770
1	14	1517	dentist	1193435061
2	14	5166	Cambodia	1170560997
3	14	6118	Russian	1170626366
4	14	6178	forgettable	1141391765

Movie Metadata

	movieId	title	genres
0	0	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	1	Jumanji (1995)	Adventure|Children|Fantasy
2	2	Grumpier Old Men (1995)	Comedy|Romance
3	3	Waiting to Exhale (1995)	Comedy|Drama|Romance
4	4	Father of the Bride Part II (1995)	Comedy

Recommendation Engine in Matrix Factorization

Concept of Latent Factor

U: User, M: Movie, R: Rating(1 ~ 5)

	M1	M2	M3	M4	M5
U1	3	5			1
U2			2	4	
U3	5		1		
U4		3		3	3

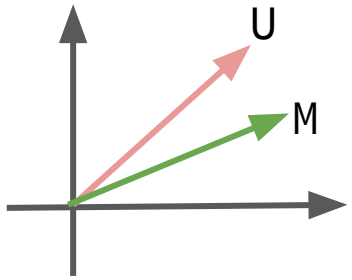
≈

U1	
U2	
U3	
U4	

X

M1	M2	M3	M4	M5

latent factor



User與Movie的距離

	M2				
U1	?	?	?	×	?
					?
					?

≈ 5

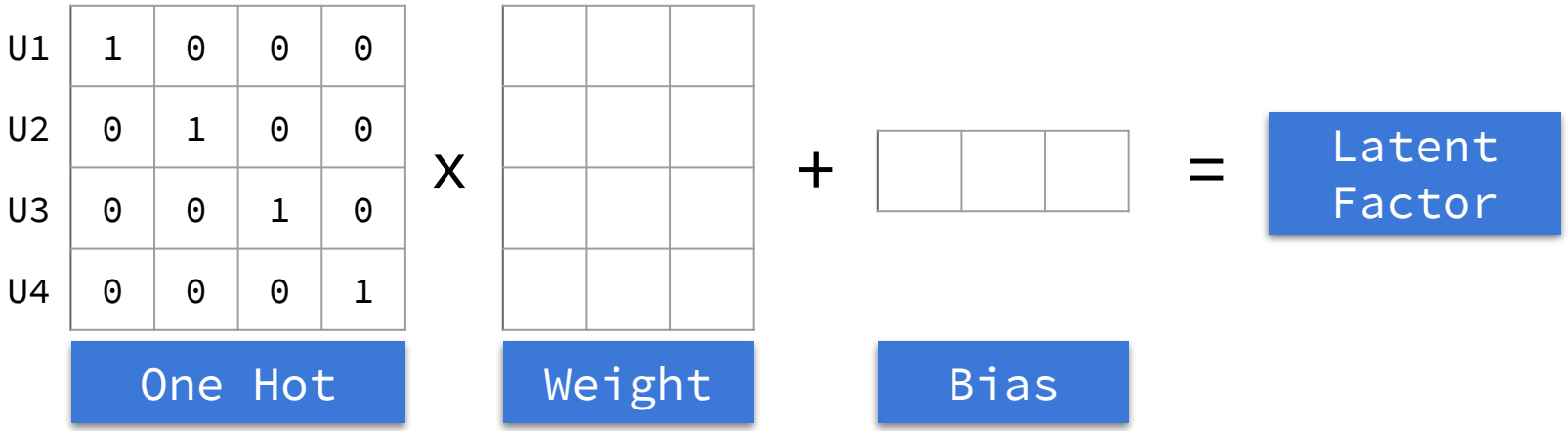
如何將user與movie
映射到向量空間?

Recommendation Engine in Matrix Factorization

Relections(Transformation)

user id ⇒ 1263, 1080, 87…
movie id ⇒ 103, 554, 187…

- 把user和movie看成是categorical variable ⇒ One Hot Encoding
- **Recall logistic regression feature transformation(Cascading Logistic Regression)**

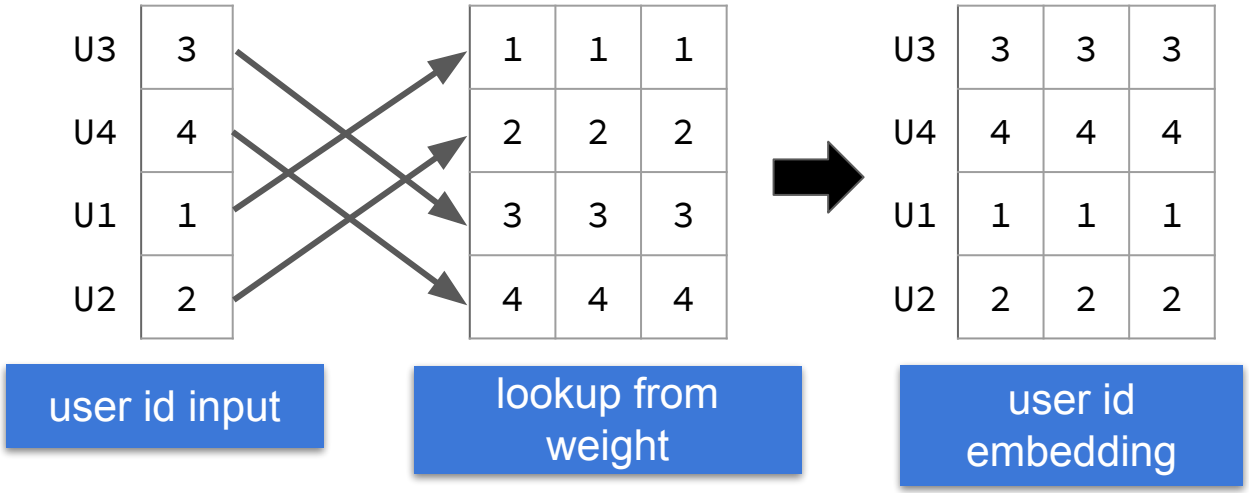


Recommedation Engine in Matrix Factorization

User ID To Embedding

U1	1	0	0	0		1	1	1		1	1	1
U2	0	1	0	0	x	2	2	2	=	2	2	2
U3	0	0	1	0		3	3	3		3	3	3
U4	0	0	0	1		4	4	4		4	4	4
	One Hot					Weight						

第i號user的latent factor等價於取Weight的第i列 ⇒ **Embedding**



- movie id embedding 也是如法炮製
- 實際都會省略 bias

Model Formulation and Loss Function

U, M ⇒ 透過embedding取得的latent factor

	M1	M2	M3	M4	M5
U1	3	5			1
U2			2	4	
U3	5		1		
U4		3		3	3

$r_{12} \Rightarrow u_1 \cdot m_2 \approx 5$

$r_{33} \Rightarrow u_3 \cdot m_3 \approx 1$

$r_{23} \Rightarrow u_2 \cdot m_3 \approx 2$

$\vdots$

Regression

$$L = \sum_{(i,j)} (u_i \cdot m_j - rij)^2$$

- User rating behavior and movie being rated are highly **biased**
- Recall linear combination ⇒ **wx + b**

$$\Rightarrow u_i \cdot m_j + b_u + b_m + b_{global}$$

- 不是U和M的外在影響因素
- Model bias

Minimizing $L = \sum_{(i,j)} (u_i \cdot m_j + b_u + b_m + b_{global} - r_{ij})^2$

Formulation

Recommendation Engine in Matrix Factorization

Prediction

以向量運算表示 $u_i \cdot m_j + b_u + b_m + b_{global}$

$Predict =$ $u_i \cdot m_j$

	M1	M2	M3	M4	M5
U1	3.5	4.7	0.87	1.87	0.8
U2	4.87	5.6	2.1	3.87	4.57
U3	5.87	4.78	1.87	1.2	3.6
U4	1.7	3.2	2.9	2.87	2.6

+ b_u (Row Wise)

0.1
0.2
0.3
0.4

+ 0.5 b_{global} (Element Wise)

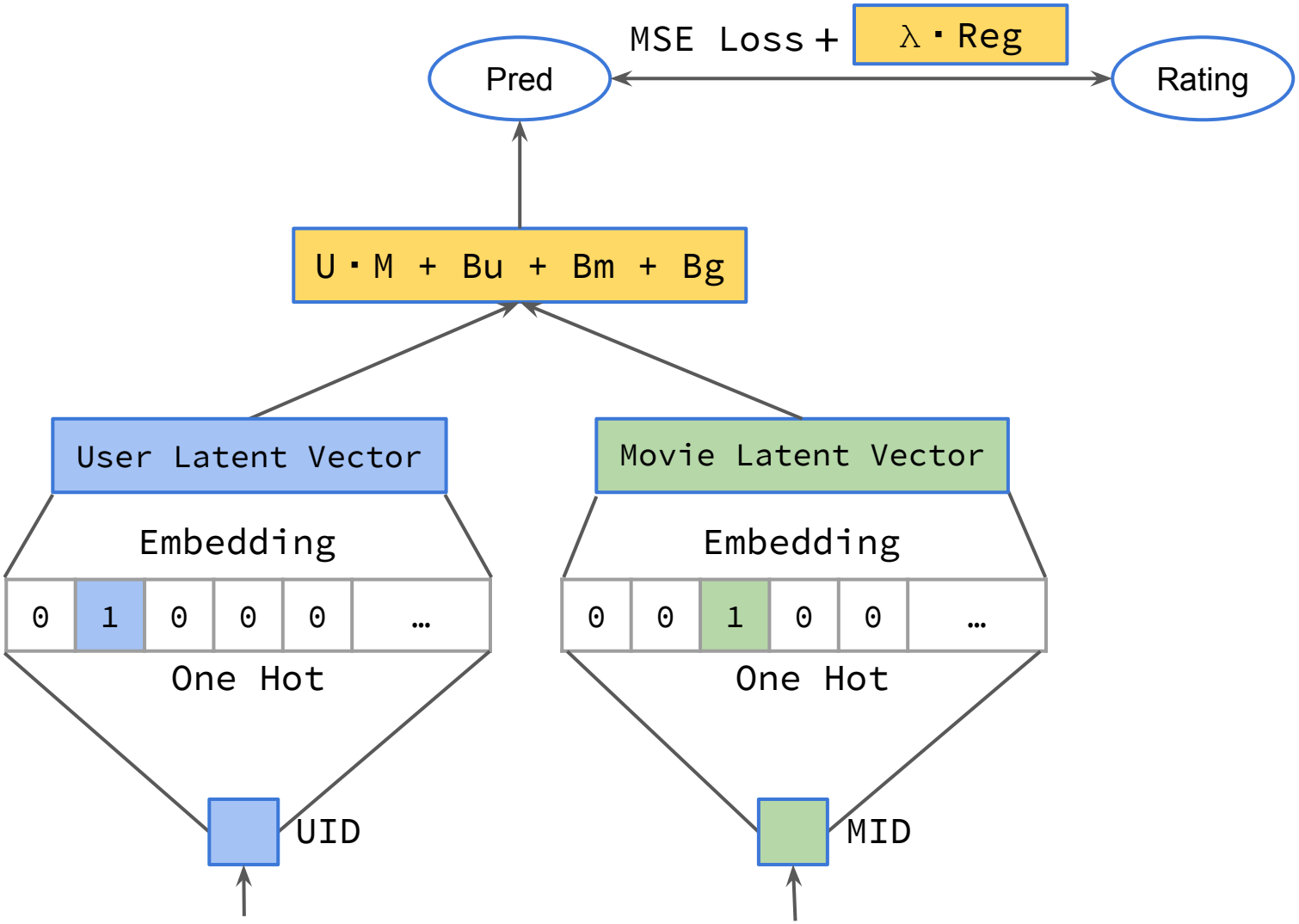
+ b_m (Column Wise)

0.8	0.7	0.6	0.5	0.4

```
[[ 4.9   6.    2.07  2.97  1.8 ]
 [ 6.37  7.    3.4   5.07  5.67]
 [ 7.47  6.28  3.27  2.5   4.8 ]
 [ 3.4   4.8   4.4   4.27  3.9 ]]
```

這個例子實際跑出來的

First Model - Basic MF



First Model - Basic MF

- Function set formulation

$$f_{u,m}(x) = u_i \cdot m_j + b_u + b_m + b_{global}$$

- Loss function

$$L = \sum (f_{u,m}(x) - r_{ij})^2 + \lambda \cdot \frac{1}{2}(\sum u^2 + \sum m^2)$$

No Bias!

Basic MF

filename	reco_model_mf.ipynb
lab filename	**lab_reco_model_mf.ipynb**
number of users	671
number of movies	9125
rating range	0.5 ~ 5分, ❑ 4分以上算正評 ❑ 未滿4分認為是負評或是不列入推薦名單
neural network	matrix factorization

Basic MF - Predict(Inference)

- ModelMF class
  - __init__: build graph
  - fit: training
    - trainGen: train data generator
    - testGen: valid data generator
    - nEpoch: number of epochs
    - **reset: if reset model checkpoint**
  - predict: predict all rating for movies of user
    - **user(list): user id list for predict**
  - epochLoss: calculate epoch loss(valid)

Recommendation Engine in Matrix Factorization

Basic MF

Traing valid data 已經幫各位處理好了，只需要載入即可

```
tr = pd.read_csv("./data/ml-latest-small/movielens.tr.csv")
te = pd.read_csv("./data/ml-latest-small/movielens.te.csv")
```

Basic MF

Data Function

1. batch產生資料, 每筆資料有三個欄位 user id, movie id, rating

```
def dataFn(data, n_batch=256, shuffle=False):
    data = data.copy()
    # 讓rating分數只有1 or 0(喜歡 or 不喜歡)
    # data["rating"] = data.rating.map(lambda e: e >= 4).astype(int)
    def _dataFn():
        idx = utils.get_minibatches_idx(len(data), n_batch, shuffle=shuffle)
        for ind in idx:
            rows = data.iloc[ind]
            yield rows.userId.values, rows.movieId.values, rows.rating.values
    return _dataFn
```

```
users:
 [0 0 0 0 0 0 0 0 0 0]
items:
 [ 931 1515   30  833  859  906 1017 1041 1047 1083]
ratings:
 [ 4.   4.   2.5  3.   3.   2.   2.   2.   2.   3.5]
```

Build Graph - Placeholder

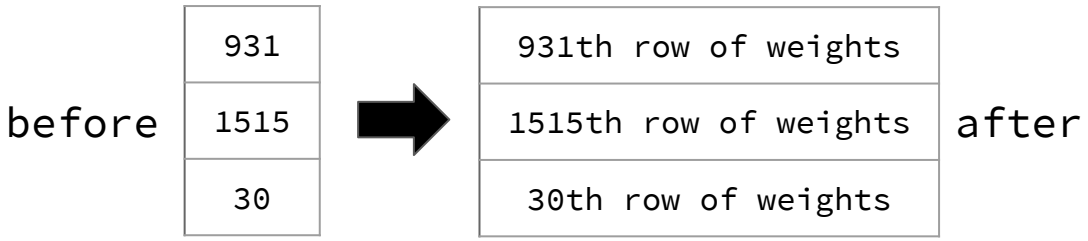
```
with tf.variable_scope("inputs"):
    self.isTrain = tf.placeholder(tf.bool, None)
    # user data
    self.user_batch = tf.placeholder(tf.int32, shape=[None], name="id_user")
    # item data
    self.item_batch = tf.placeholder(tf.int32, shape=[None], name="id_item")
    # labels
    self.rate_batch = tf.placeholder(tf.float32, shape=[None])
```

- user_batch:
  - [None] ⇒ None代表變動的batch size
- item_batch:
  - [None] ⇒ None代表變動的batch size
- rate_batch:
  - [None] ⇒ None代表變動的batch size
- isTrain指示是否是training狀態

Build Graph – Embedding

```
with tf.variable_scope("embedding"):
    # User and item bias variables
    self.w_bias_user = tf.Variable(initFn(shape=[user_num]), name="embd_bias_user")
    self.w_bias_item = tf.Variable(initFn(shape=[item_num]), name="embd_bias_item")
    self.w_user = tf.Variable(initFn(shape=[user_num, dim]), name="embd_user")
    self.w_item = tf.Variable(initFn(shape=[item_num, dim]), name="embd_item")
    # ------------------------------------------------------------------------------
    # embedding_lookup: Looks up 'ids' in a list of embedding tensors
    # Bias embeddings for user and items, given a batch
    # shape of self.bias_user = (user_batch,)
    # shape of self.bias_item = (item_batch,)
    self.bias_user = tf.nn.embedding_lookup(self.w_bias_user, self.user_batch, name="bias_user")
    self.bias_item = tf.nn.embedding_lookup(self.w_bias_item, self.item_batch, name="bias_item")
    # Weight embeddings for user and items, given a batch
    # shape of self.embd_user = (user_batch * dim)
    # shape of self.embd_item = (item_batch * dim)
    self.embd_user = tf.nn.embedding_lookup(self.w_user, self.user_batch, name="embedding_user")
    self.embd_item = tf.nn.embedding_lookup(self.w_item, self.item_batch, name="embedding_item")
```

- `tf.nn.embedding_lookup(weights, index)`



Build Graph - Predict(Inference)

```
with tf.variable_scope("computation"):
    # self.embd_user = tf.layers.dropout(self.embd_user, rate=0.5, training=self.isTrain)
    # self.embd_item = tf.layers.dropout(self.embd_item, rate=0.5, training=self.isTrain)
    infer = tf.reduce_sum(tf.multiply(self.embd_user, self.embd_item), 1)
    infer = tf.add(infer, self.bias_global)
    infer = tf.add(infer, self.bias_user)
    self.infer = tf.add(infer, self.bias_item, name="infer")
```

- infer $\Rightarrow u_i \cdot m_j + b_u + b_m + b_{global}$

Build Graph - Loss

```
with tf.variable_scope("loss"):
    self.regularizer = reg * tf.add(tf.nn.l2_loss(self.embd_user), tf.nn.l2_loss(self.embd_item))
    # l2_loss: Computes half the L2 norm of a tensor without the sqrt => sum(t ** 2) / 2
    # self.loss = tf.nn.l2_loss(self.infer - self.rating[:, tf.newaxis]) + self.regularizer
    self.loss = tf.losses.mean_squared_error(labels=self.infer, predictions=self.rate_batch) +
                self.regularizer

    # self.loss = tf.add(cost_l2, tf.multiply(self.regularizer, penalty))
    # for eval
    self.rmse_loss = tf.sqrt(
        tf.reduce_mean(
            tf.losses.mean_squared_error(
                labels=self.rate_batch, predictions=self.infer)), name="rmse_loss")
    self.mae_loss = tf.reduce_mean(tf.abs(self.infer - self.rate_batch), name="mae_loss")

with tf.variable_scope("train"):
    self.train_op = tf.train.AdagradOptimizer(learning_rate).minimize(self.loss)
```

- loss: mean squared loss with regularizer term
  - ex: tf.nn.l2_loss(self.embd_user) + …
- rmse_loss: root mean squared error(for evaluate)
  - Netflix的評分標準
- mae_loss: mean absolute error(for evaluate)

Basic MF - Training

Trainging

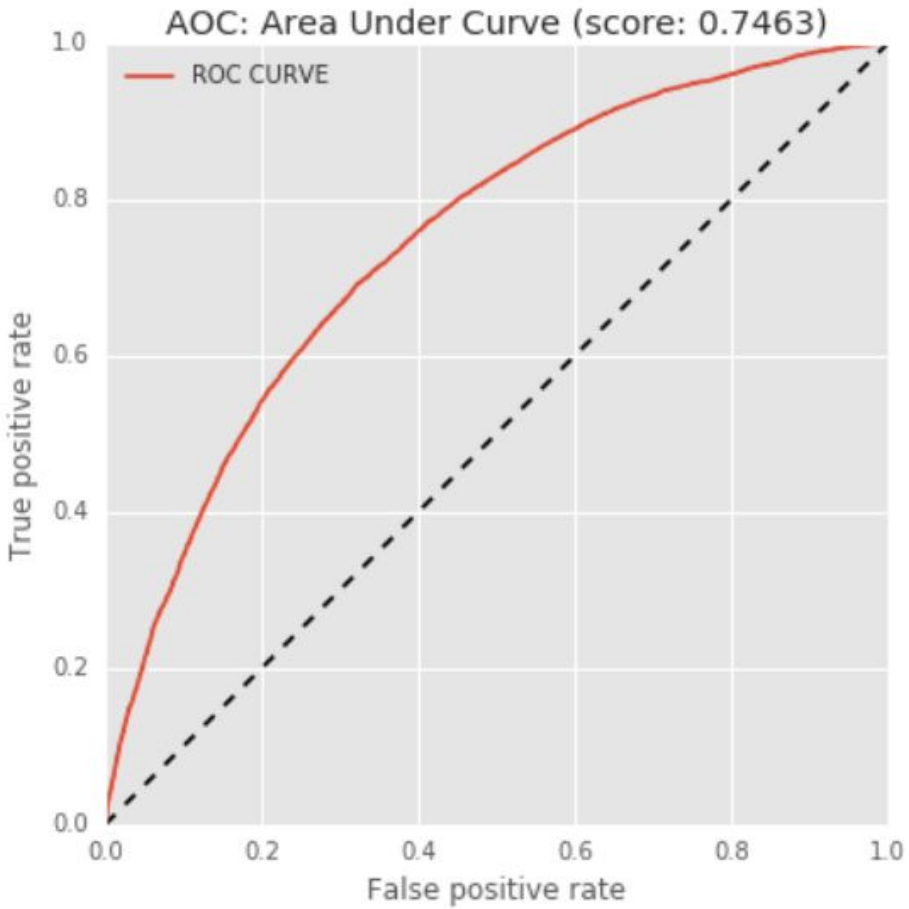
```
n_batch = 128
with tf.Session(graph=model.graph,) as sess:
    model.fit(sess, dataFn(tr, n_batch=n_batch), dataFn(te, n_batch=n_batch), nEpoch=20, reset=True)
```

```
reset model: clean model dir: ./model/model_mf_1 ...
Epoch    Train Error     Val Error       Elapsed Time
01       1.130           1.005           1.514 secs, saving ...
02       0.987           0.982           1.323 secs, saving ...
03       0.956           0.969           1.429 secs, saving ...
04       0.936           0.960           1.680 secs, saving ...
05       0.922           0.953           1.350 secs, saving ...
06       0.911           0.948           1.685 secs, saving ...
07       0.903           0.943           1.746 secs, saving ...
08       0.896           0.940           1.587 secs, saving ...
09       0.890           0.937           1.548 secs, saving ...
10       0.885           0.935           1.747 secs, saving ...
11       0.880           0.933           1.727 secs, saving ...
12       0.877           0.931           1.510 secs, saving ...
13       0.873           0.930           1.494 secs, saving ...
14       0.870           0.929           1.379 secs, saving ...
15       0.868           0.927           1.366 secs, saving ...
16       0.865           0.927           1.354 secs, saving ...
17       0.863           0.926           1.361 secs, saving ...
18       0.861           0.925           1.339 secs, saving ...
19       0.859           0.924           1.378 secs, saving ...
20       0.857           0.924           1.471 secs, saving ...
```

After 20 epochs,
valid RMSE about 0.92

若發現到valid loss增加則不會儲存!

Basic MF - ROC AUC



TP Rate cross FP Rate:

AUC = 0.5 (no discrimination 無鑑別力)
0.7 ≦ AUC ≦ 0.8 (可接受的鑑別力)
0.8 ≦ AUC ≦ 0.9 (優良的鑑別力)
0.9 ≦ AUC ≦ 1.0 (極佳的鑑別力)

實際上真的如此嚴格? 非0.8就不算優良?

Basic MF – Google Wide and Deep Model

Google Play App上推薦系統Model分數

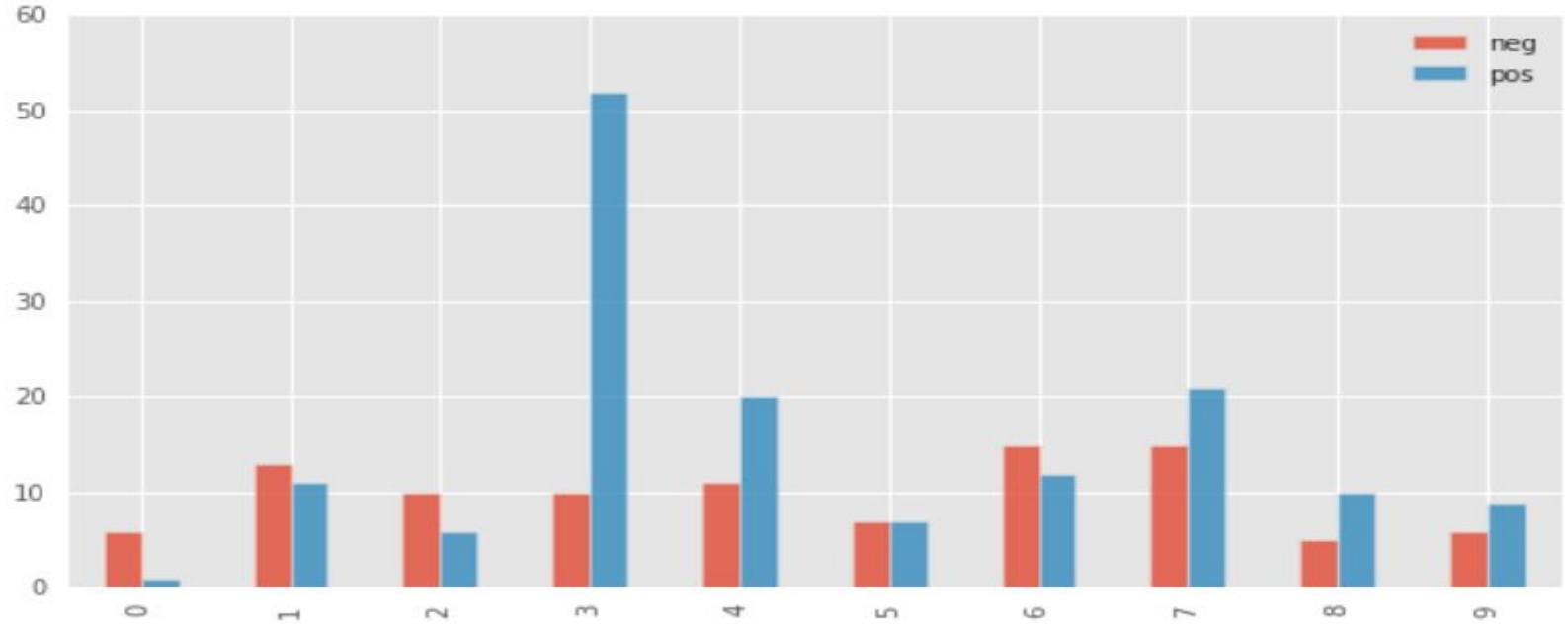
Table 1: Offline & online metrics of different models. Online Acquisition Gain is relative to the control.

Model	Offline AUC	Online Acquisition Gain
Wide (control)	0.726	0%
Deep	0.722	+2.9%
Wide & Deep	0.728	+3.9%

Paper: Wide & Deep Learning for Recommender Systems

- AUC: 0.728的Model也能夠帶來實際收益
  - 考慮系統規模、user feedback困難度等等...

Basic MF - Precision at K(10)



- 0號, 2號, 5號, 9號 user 正向評價數量 < 10, 就算model全部預測命中, 命中率也不會是 100%! ex: 0號user只有1個正向評價, 全部命中也只得到0.1的分數
- 3號user正向評價是負向評價的5倍多, 就算亂猜, 中的機率也很高

Recommendation Engine in Matrix Factorization

Basic MF - Precision at K(10)

```
def strict_condition(label):
    label = label[label != 0]
    pos, neg = sum(label >= 4), sum(label < 4)
    return len(label) >= 10 and pos <= neg and pos > 0

print("rating數量 >= 10 且 負評價數量 >= 正評價數量 有 [{}] 人".format(sum(strict_condition(label) for label in teRatingMat)))

def norm_condition(label):
    label = label[label != 0]
    return sum(label >= 4) > 0 and sum(label < 4) > 0

print("rating正評價數量 >= 0 且 rating負評價數量 >= 0 有 [{}] 人".format(sum(norm_condition(label) for label in teRatingMat)))
```

```
rating數量 >= 10 且 負評價數量 >= 正評價數量 有 [209] 人
rating正評價數量 >= 0 且 rating負評價數量 >= 0 有 [669] 人
```

- strict condition: rating數量 >= 10 且 負評數 >= 正評數 (209人)
- normal condition: 正評數 > 0 且 負評數 > 0

```
strict condition precision at 10:  0.506220095694
norm condition precision at 10:  0.632286995516
```

Don't panic!

對照組

實際上資料量夠大可以忽略Outliter!

Recommendation Engine in Matrix Factorization

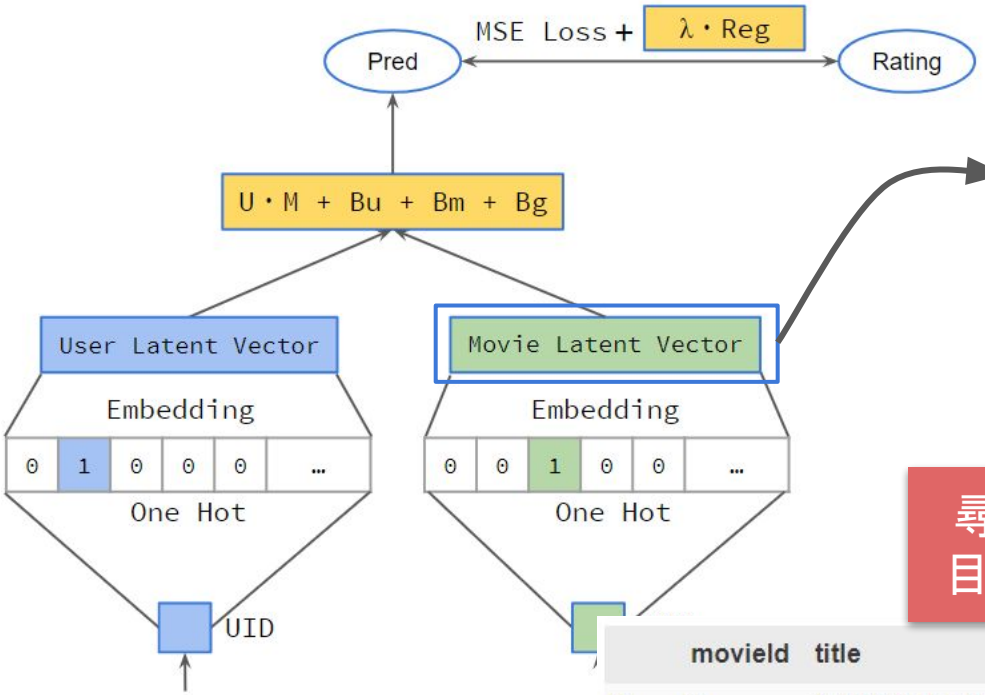
Basic MF – NDCG at K(10)

```
strict condition ndcg at 10:   0.633302954025
norm condition ndcg at 10:   0.760044561361
```

- 0.6分以上及格
- 已經比前一個memory base進步

Recommendation Engine in Matrix Factorization

Basic MF – 活用Latent Factor(Embedding)!



使用Latent Factor來找出相似的電影

尋找與Toy Story相似的電影
目前來看似乎結果不太合理...

	movieId	title	genres
0	0	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
7551	7551	Letters to Juliet (2010)	Drama|Romance
5158	5158	Laws of Attraction (2004)	Comedy|Romance
8473	8473	Lone Survivor (2013)	Action|Drama|Thriller|War
900	900	Perfect Candidate, A (1996)	Documentary
2786	2786	My Chauffeur (1986)	Comedy
6867	6867	To the Left of the Father (Lavoura Arcaica) (2...	Drama
3794	3794	From Hell (2001)	Crime|Horror|Mystery|Thriller
921	921	T-Men (1947)	Film-Noir
4279	4279	The Pumaman (1980)	Action|Adventure|Fantasy|Sci-Fi

Model All Metrics

Model	Dummy Model(亂猜)	Basic MF
RMSE Loss	3.23	0.92
ROC AUC	0.50	0.74
NDCG	strict: 0.45 normal: 0.63	strict: 0.63 normal: 0.76
Precision at 10	strict: 0.37 normal: 0.54	strict: 0.50 normal: 0.63

Recommendation Engine in Matrix Factorization

Lab: lab_reco_model_mf.ipynb (20 minutes)

搜尋 ”Do:” 就可以找到可能要修改的位置

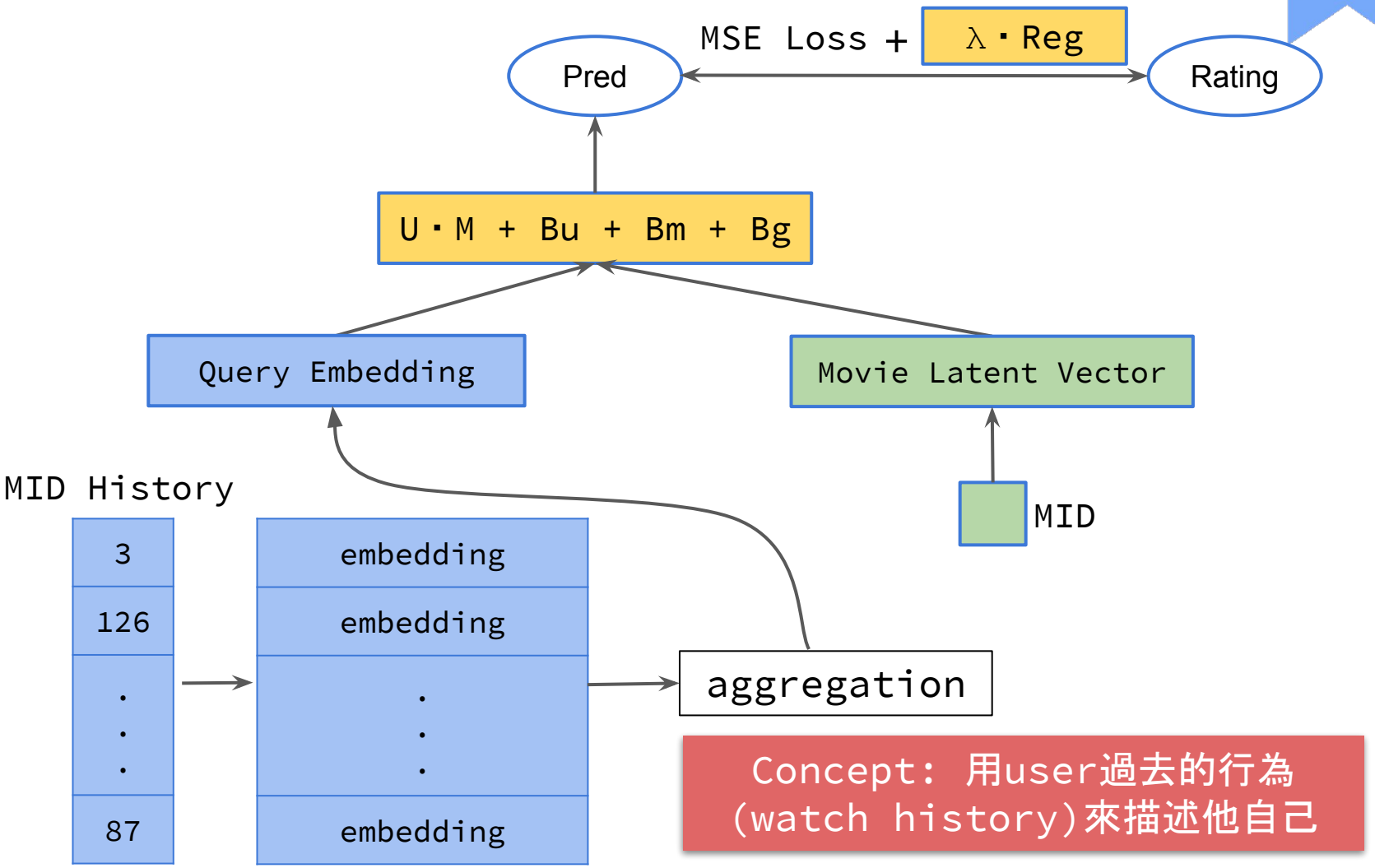
lab filename	lab_reco_model_mf.ipynb
target	您可能要修改的參數: ❑ learning rate ❑ dim: dimension of latent factor(user and movie) ❑ reg: 正規項的強度 ❑ dropout ❑ loss function ❑ optimizer
ideal score	valid loss: 將低到0.92左右 roc auc: 0.74左右 ndcg: strict condition 0.63左右

Model - MF with History

- 如果新的User進來…
  - 假設Profile資訊不足，不知道是男是女…
    - 推薦最受歡迎的電影(完全沒User rating)
  - 就算累積了Rating history，Model還是要重train
    - Fall back to content base recommendation

在不重train的狀況下，希望能夠對新來的user(有rating history)做推薦

Model - MF with History



Model - MF with History

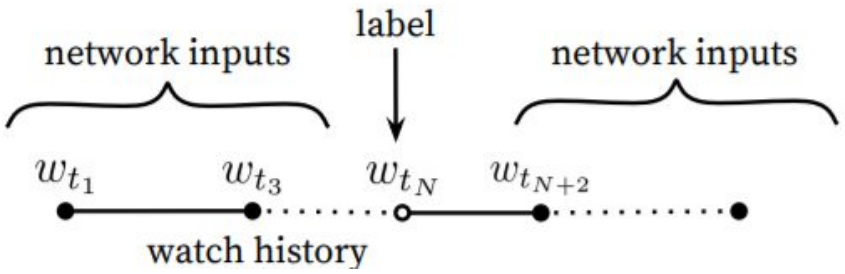
filename	reco_model_mf_with_history.ipynb
lab filename	**lab_reco_model_mf_with_history.ipynb**
number of users	671
number of movies	9125
rating range	0.5 ~ 5分, ❑ 4分以上算正評 ❑ 未滿4分認為是負評或是不列入推薦名單
neural network	matrix factorization

Model - MF with History

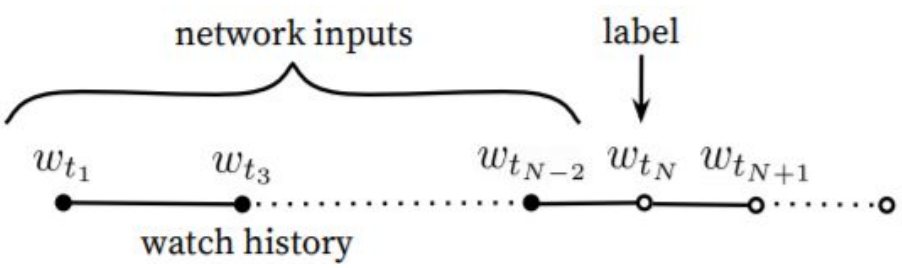
	user_id	query_movie_ids	query_movie_ids_len	candidate_movie_id	rating
0	47	[263, 2860, 3856, 6034, 2409, 2374, 7128, 5339...	357	1393	4.5
1	175	[966, 5026, 4395, 6042, 3871, 6521, 5020, 953,...	177	4002	3.0
2	261	[45, 1104, 154, 4514, 4171, 3801, 3727, 3644, ...	471	3089	2.0

- 觀察47號user
  - query_movie_ids: train data中47號的有rating的movie id, 排除1393號 ⇒ leave one out (這裡的設計也可以只用47號喜歡的電影)

leave one out for predict



leave one out(考慮時間)



Model - MF with History

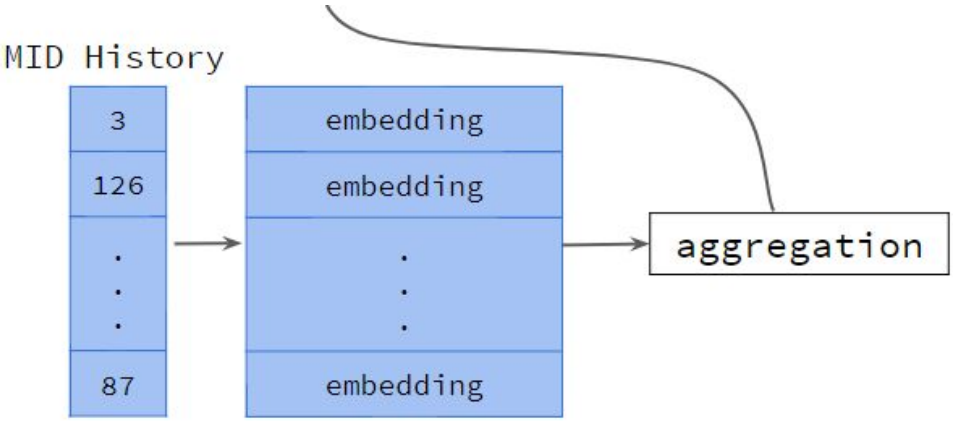
	user_id	query_movie_ids	query_movie_ids_len	candidate_movie_id	rating
0	47	[263, 2860, 3856, 6034, 2409, 2374, 7128, 5339...	357	1393	4.5
1	175	[966, 5026, 4395, 6042, 3871, 6521, 5020, 953,...	177	4002	3.0
2	261	[45, 1104, 154, 4514, 4171, 3801, 3727, 3644, ...	471	3089	2.0

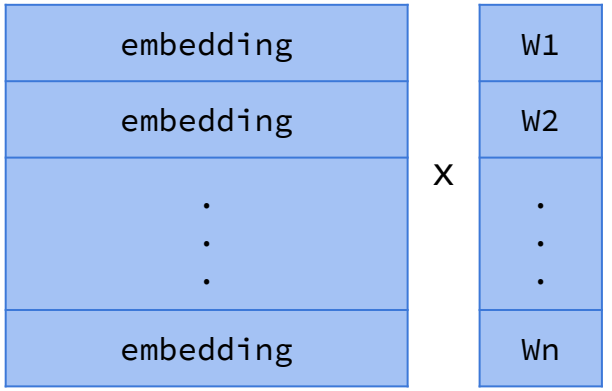
- 觀察47號user
  - query_movie_ids: train data中47號的有rating的movie id, 排除1393號 ⇒ leave one out (這裡的設計也可以只用47號喜歡的電影)
  - query_movie_ids_len: 描述query_movie_ids movie id的個數 (tensorflow限制, mini batch裡面的變數長度必須要一致)

所以47號User在train data裡只有一筆資料了?

Recommendation Engine in Matrix Factorization

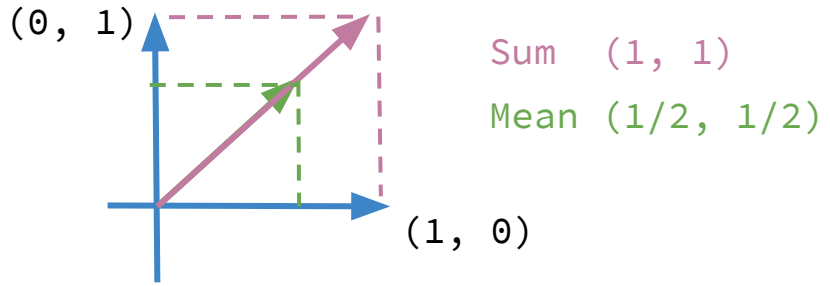
Movie ID Aggregation(Pooling)?





Tensorflow documentation上找到處理multivalent embedding的方式	
Sum	embedding向量相加
Mean	embedding向量相加後平均
SqrtN	將weight normalize後對embedding做weighted sum

效果較好



Recommendation Engine in Matrix Factorization

MF with History - Aggregation(Pooling)

```
with tf.variable_scope("computation"):
    # weighted avg of query embedding
    '''sqrtn aggregation(pooling), X: data, W: weight
           X_1*W_1 + X_2*W_2 + ... + X_n*W_n / sqrt(W_1**2 + W_2**2 + ... W_n**2)
         = weighted sum of X and normalized W
       here data = self.query_emb, weight = weighted '''
    query_emb_mask = tf.sequence_mask(self.query_movie_ids_len)
    weighted = tf.nn.l2_normalize(tf.to_float(query_emb_mask), 1)[:, :, tf.newaxis]
    self.query_emb = tf.reduce_sum(self.query_emb * weighted, 1)
    self.query_bias = tf.matmul(self.query_emb, self.b_query_movie_ids[:, tf.newaxis])
```

```
[2]    [1, 1, 0, 0]    [0.7, 0.7,   0, 0]        [array, array,     0, 0]
[3]  ➡ [1, 1, 1, 0]  ➡ [0.6, 0.6, 0.6, 0]   X    [array, array, array, 0]
[1]    [1, 0, 0, 0]    [  1,   0,   0, 0]        [array,     0,     0, 0]
```

query_movie_ids_len tf.sequence_mask normalize

Weighted Sum!

MF with History - Prediction

```
infer = tf.reduce_sum(self.query_emb * self.candidate_emb, 1, keep_dims=True)
infer = tf.add(infer, self.b_global)
infer = tf.add(infer, self.query_bias)
self.infer = tf.add(infer, self.candidate_bias, name="infer")
```

```
# one query for all items
self.pred = tf.matmul(self.query_emb, tf.transpose(self.w_candidate_movie_id)) + \
            tf.reshape(tf.matmul(self.w_candidate_movie_id,
                                 self.b_candidate_movie_id[:, tf.newaxis]), (1, -1)) + \
            self.query_bias + \
            self.b_global
```

- infer節點與pred節點做的事情是一樣的，都是帶下列的公式

$$f_{u,m}(x) = u_i \cdot m_j + b_u + b_m + b_{global}$$

- pred節點純粹是為了執行效能而已

MF with History - Metrics and Comparison

Model	MF	MF with history
RMSE Loss	0.92	0.92
ROC AUC	0.74	0.74
NDCG	strict: 0.63 normal: 0.76	strict: 0.63 normal: 0.76
Precision at 10	strict: 0.50 normal: 0.63	strict: 0.51 normal: 0.63

- MF with history的Model效果沒有進步
- 但後者的Model較有彈性!

Lab: lab_reco_model_mf_with_history.ipynb (20 minutes)

搜尋 ”Do:” 就可以找到可能要修改的位置

lab filename	lab_reco_model_mf_with_history.ipynb
target	您可能要修改的參數: ❑ learning rate ❑ dim: dimension of latent factor(user and movie) ❑ reg: 正規項的強度 ❑ dropout ❑ loss function ❑ optimizer
ideal score	valid loss: 降低到0.92左右 roc auc: 0.74左右 ndcg: strict condition 0.63左右

Recommendation Engine with DNN

- Machine Learning Framework
- TensorFlow Fundamentals
- Deep Neural Network(DNN) Tips
- Recommedation Engine in Matrix Factorization
- Matrix Factorization with DNN

Model - MF with DNN

- 同樣的想法，也來改進一下**Movie Latent Factor**

	userId	movieId	title	genres	rating
0	0	30	Dangerous Minds (1995)	Drama	2.5
1	0	833	Dumbo (1941)	Animation|Children|Drama|Musical	3.0
2	0	859	Sleepers (1996)	Thriller	3.0
3	0	906	Escape from New York (1981)	Action|Adventure|Sci-Fi|Thriller	2.0
4	0	931	Cinema Paradiso (Nuovo cinema Paradiso) (1989)	Drama	4.0

- Movie Meta好像太少了點
  - title欄位有**年份資料**
  - genres告訴我們一部電影可以有多種類型
  - rating是user對電影的評分，一部電影被多個user評分
    ⇒ 可以算出**平均評分**

Model – MF with DNN

- 於是電影可以用下列Meta來描述

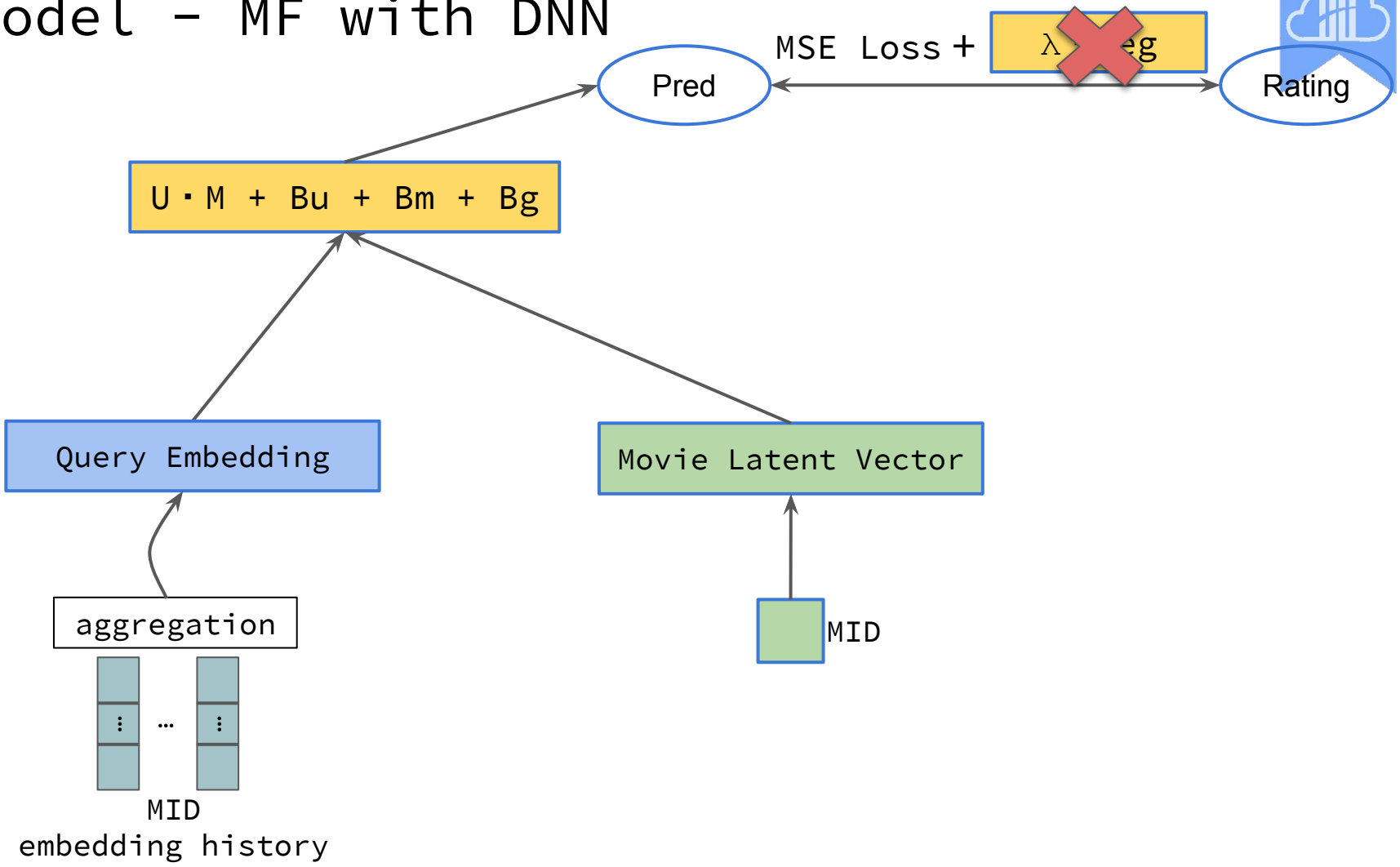
genres	genres_len	avg_rating	year	candidate_movie_id	rating
[1, 0, 3, 0]	3	0.716049	0.833333	1393	4.5
[5, 13, 10, 1]	4	0.643739	0.877193	4002	3.0
[0, 7, 2, 0]	3	0.545752	0.859649	3089	2.0
[7, 11, 2, 0]	3	0.448148	0.833333	1308	3.0
[1, 0, 0, 0]	1	0.779449	0.736842	2340	1.5

- genres: multivalent variable ⇒ embedding ⇒ average
- genres_len: 描述genres長度
- avg_rating: 電影的平均評分(已normalize to 0 ~ 1之間)
- year: 從title extract出來(已normalize to 0 ~ 1之間)

Enrich movie metadata
- 還可以Crawler搜尋電影導演、演員、票房等等...

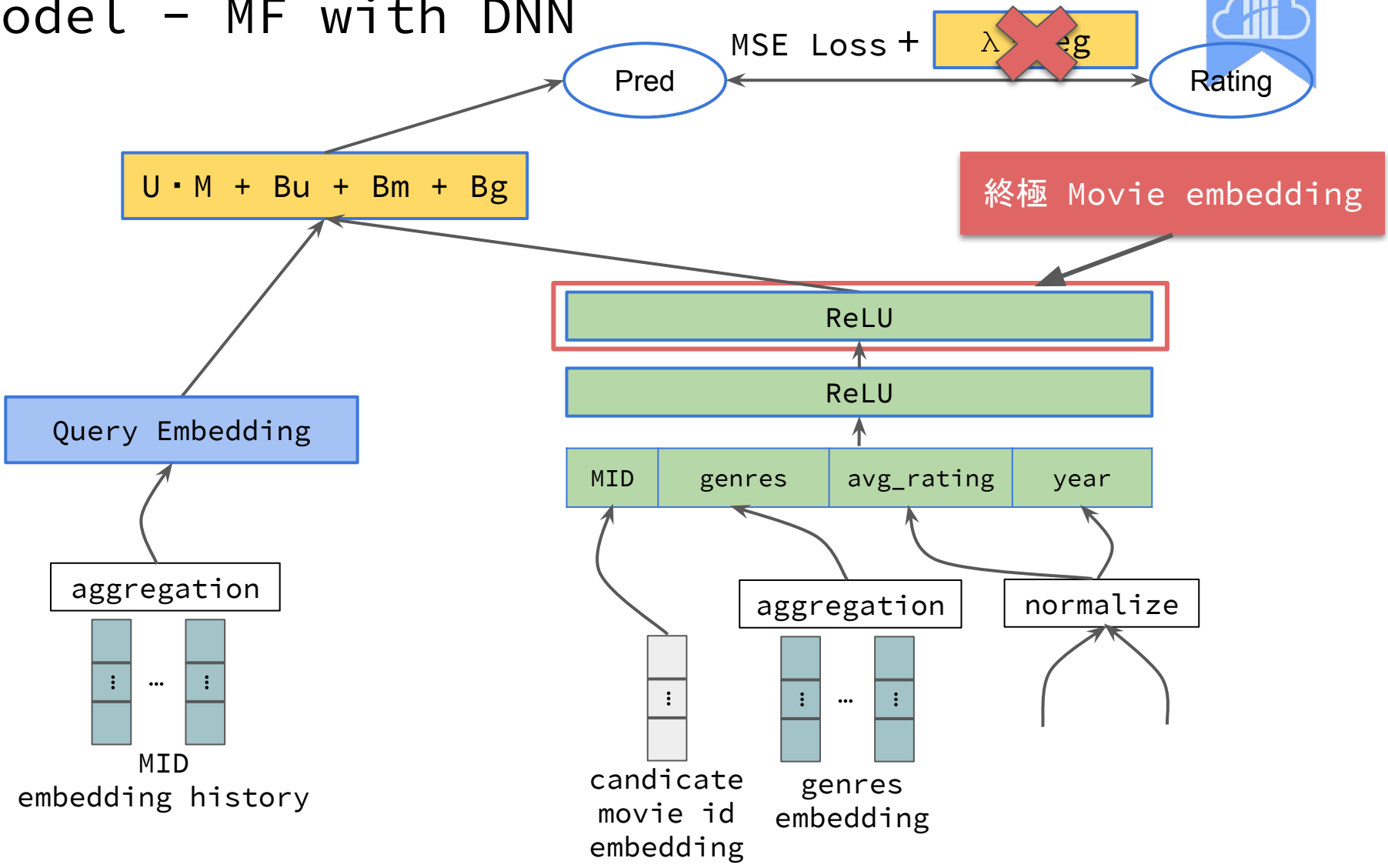
Recommendation Engine with DNN

Model – MF with DNN



Recommendation Engine with DNN

Model - MF with DNN



Model - MF with DNN

filename	reco_model_mf_dnn.ipynb
lab filename	**lab_reco_model_mf_dnn.ipynb**
number of users	671
number of movies	9125
rating range	0.5 ~ 5分, ❑ 4分以上算正評 ❑ 未滿4分認為是負評或是不列入推薦名單
neural network	matrix factorization with dnn

Recommendation Engine with DNN

Model – MF with DNN(Movie Encoding)

```
with tf.variable_scope("dnn"):
    # encode [item embedding + item metadata]
    self.item_repr = tf.concat([self.candidate_emb, self.genres_emb, self.avg_rating[:, tf.newaxis], self.year[:, tf.newaxis]], 1)
    self.candidate_bias = tf.matmul(self.item_repr, self.b_candidate_movie_id[:, tf.newaxis])

    # dp_scale = 0.5
    self.item_repr = tf.layers.dense(self.item_repr, dim, kernel_initializer=init_fn, activation=tf.nn.relu)
    # self.item_repr = tf.layers.dropout(self.item_repr, dp_scale, training=self.isTrain)
    self.item_repr = tf.layers.dense(self.item_repr, dim, kernel_initializer=init_fn, activation=tf.nn.relu)
    # self.item_repr = tf.layers.dropout(self.item_repr, dp_scale, training=self.isTrain)
```

- item_repr: concat metadata
  - candidate movie id embedding (16) +
  - genres embedding (8) +
  - 平均分數 (1) +
  - 年份 (1)

  ⇒ shape = 16 + 8 + 1 + 1 = 26

- 兩層DNN(fully connected layer) + ReLU ⇒ encoding
- **這裡的Network structure可以自行調整(增加層數 or 調整neuron數)**

Model - MF with DNN

Trainging

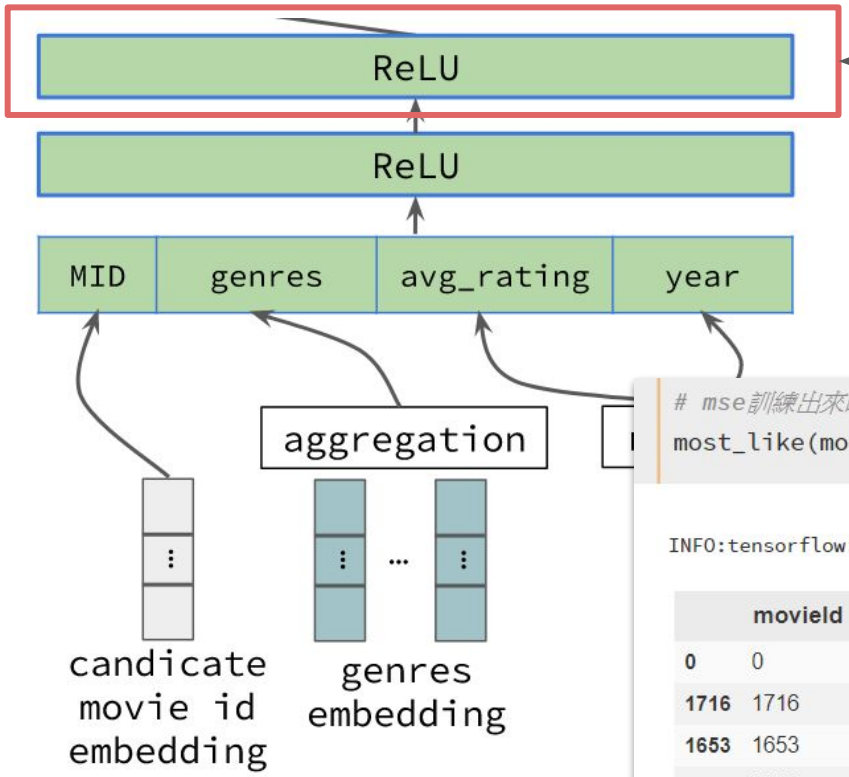
```
# !cp -a ./model/model_mf_with_dnn_bak/* ./model/mode
# 每個batch的數量
n_batch = 128
with tf.Session(graph=model.graph) as sess:
    model.fit(sess, dataFn(trProcessed, n_batch=n_bat
```

```
reset model: clean model dir: ./model/model_mf_with_dnn ...
Epoch    Train Error     Val Error       Elapsed Time
01       1.096           0.934           14.236 secs, saving ...
02       0.961           0.894           15.311 secs, saving ...
03       0.931           0.870           14.697 secs, saving ...
04       0.910           0.854           13.682 secs, saving ...
05       0.894           0.844
06       0.881           0.835
07       0.870           0.827
08       0.862           0.823           15.012 secs, saving ...
09       0.856           0.818           13.786 secs, saving ...
10       0.851           0.820           13.758 secs
```

Valid RMSE error已經可以低達0.81

Recommendation Engine with DNN

MF with DNN – 活用Laten Factor(Embedding)!



找出跟Toy Story相似的電影
- ❏ Indide Out
- ❏ Toy Story2
- ❏ Antz(螞蟻雄兵)

...

```
# mse訓練出來的model
most_like(model, 0, k=11)
```

```
INFO:tensorflow:Restoring parameters from ./model/model_mf_with_dnn/model-9
```

	movieId	title	genres
0	0	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1716	1716	NeverEnding Story, The (1984)	Adventure|Children|Fantasy
1653	1653	Return to Oz (1985)	Adventure|Children|Fantasy
8911	8911	Inside Out (2015)	Adventure|Animation|Children|Comedy|Drama|Fantasy
1815	1815	Antz (1998)	Adventure|Animation|Children|Comedy|Fantasy
3217	3217	Emperor's New Groove, The (2000)	Adventure|Animation|Children|Comedy|Fantasy
7945	7945	Asterix and the Vikings (Astérix et les Viking...	Adventure|Animation|Children|Comedy|Fantasy
2506	2506	Toy Story 2 (1999)	Adventure|Animation|Children|Comedy|Fantasy
3079	3079	The Spiral Staircase (1945)	Horror|Mystery|Thriller
813	813	Escape to Witch Mountain (1975)	Adventure|Children|Fantasy
6246	6246	MirrorMask (2005)	Adventure|Children|Drama|Fantasy

Model – 實際上使用Embedding的例子

使用User embedding找出該user偏好的衣服類型

甚至可以找不同類型的搭配(馬克思禮服找到一堆寬腿褲)

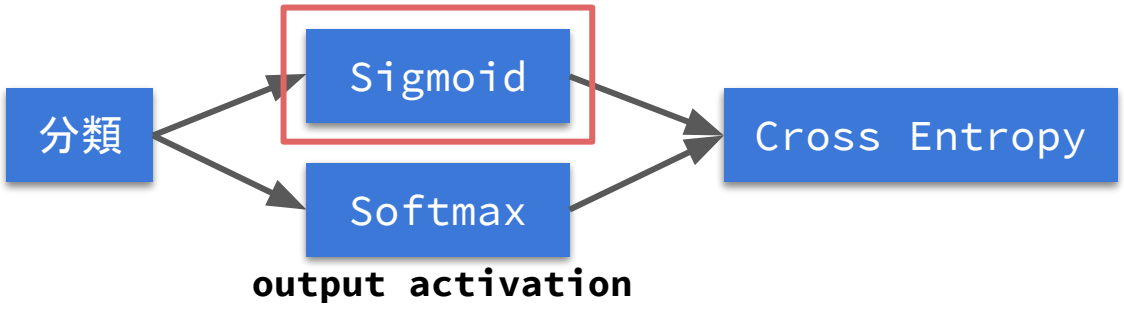
https://making.dia.com/embedding-everything-for-anything2anything-recommendations-fca7f58f53ff

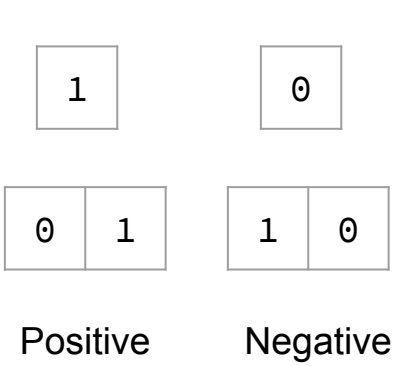
MF with DNN - Metrics and Comparison

Model	MF	MF with history	MF with DNN
RMSE Loss	0.92	0.92	**0.82**
ROC AUC	0.74	0.74	**0.80**
NDCG	strict: 0.63 normal: 0.76	strict: 0.63 normal: 0.76	**strict: 0.73** **normal: 0.86**
Precision at 10	strict: 0.50 normal: 0.63	strict: 0.51 normal: 0.63	**strict: 0.59** **normal: 0.68**

Model - MF with DNN(Classification)

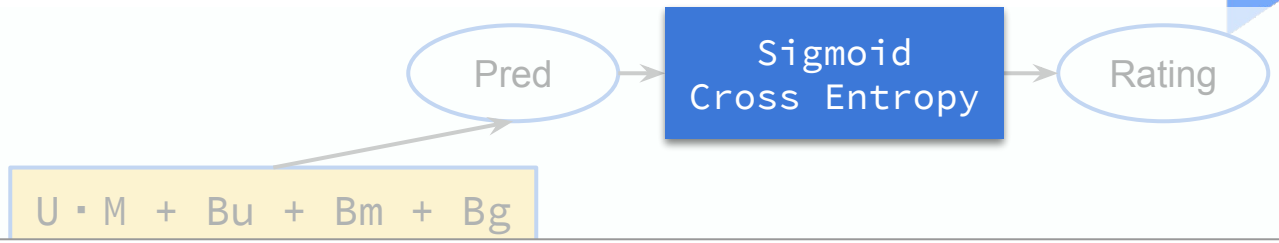
- Movie rating: explicit feedback
- **通常User不會這麼熱心的對你的商品評價**
  - implicit feedback
    - 商品點閱次數
    - Youtube影片觀看時間
    - …
    - ⇒ 通常只能得到1(Positive), 0(Negtive)
- **Label 0.5 ~ 5 ⇒ 1 or 0 (這裡4分以上 = 1 others 0)**
  - Regression ⇒ Binary classification
  - P(使用者A 喜歡 電影B) = ??





Recommendation Engine with DNN

Model - MF with DNN(Classification)

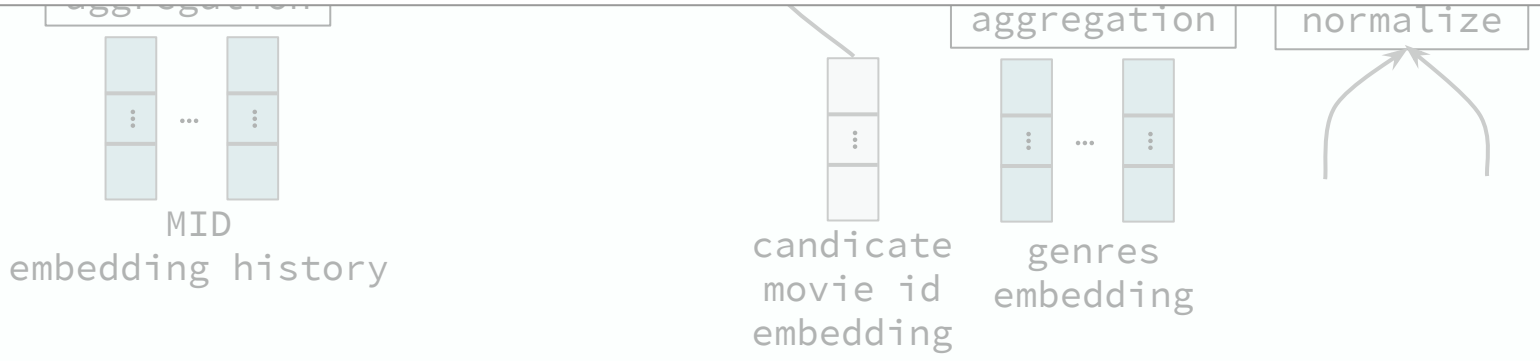


- Function set

$$f_{u,m}(x) = \sigma(u_i \cdot m_j + b_u + b_m + b_{global})$$

- Loss function: cross entropy

$$L = -\sum(\hat{y} \cdot lnf_{u,m}(x) + (1 - \hat{y})(1 - lnf_{u,m}(x)))$$



Recommendation Engine with DNN

Model – MF with DNN

```
with tf.variable_scope("loss"):
    # if rating >= 4 then 1 else 0
    self.alter_rating = tf.to_float(self.rating >= 4)
    self.loss = tf.reduce_mean(tf.nn.sigmoid_cross_entropy_with_logits(labels=self.alter_rating[:, np.newaxis], logits=self.infer))

    # for eval
    self.rmse_loss = tf.sqrt(tf.losses.mean_squared_error(labels=self.alter_rating[:, tf.newaxis], predictions=self.infer))
    self.mae_loss = tf.reduce_mean(tf.abs(self.infer - self.alter_rating[:, tf.newaxis]))
    pass
```

- alter_rating: **在tensorflow裡轉換label**
  - if rating >= 4 ⇒ 1, else 0

Model - MF with DNN

Trainging

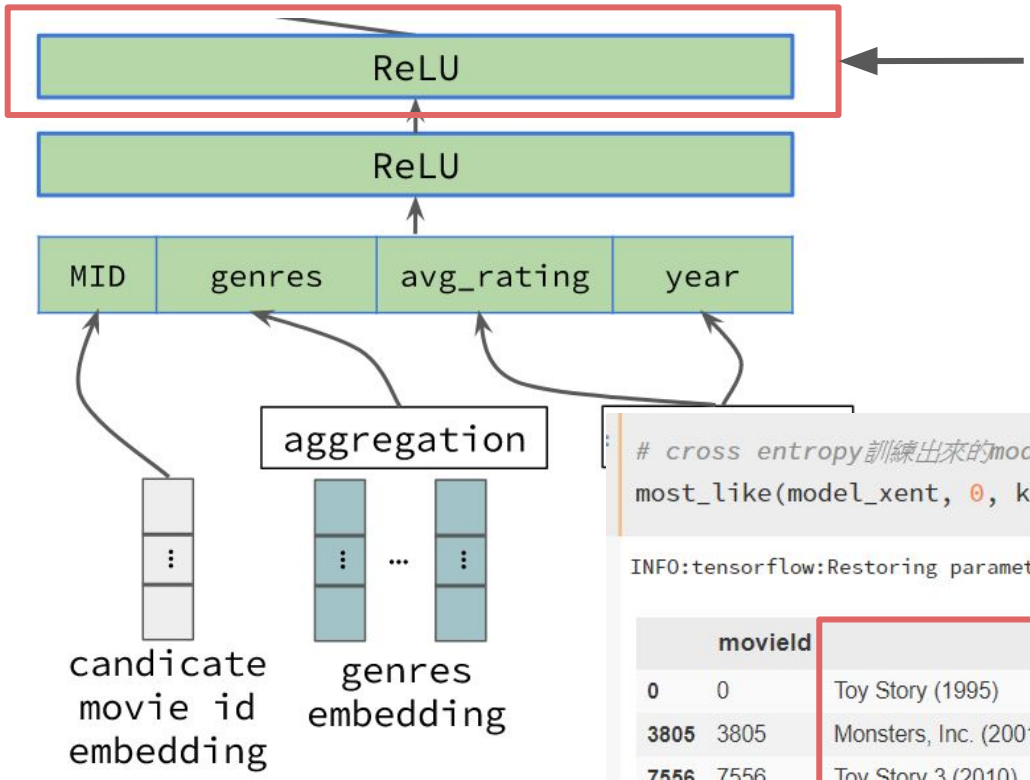
```
n_batch = 128
with tf.Session(graph=model_xent.graph) as sess:
    model_xent.fit(sess, dataFn(trProcessed, n_batch=n_batch, sh
```

```
reset model: clean model dir: ./model/model_mf_with_dnn_xent ...
Epoch   XEntropy Error  Val XENT Error  Val RMSE Error  Elapsed Time
01      0.653           0.622           0.745           15.133 secs, saving ...
02      0.607           0.590           0.888           15.839 secs, saving ...
03      0.580           0.571           0.930           15.542 secs, saving ...
04      0.562           0.559           0.986           14.085 secs, saving ...
05      0.551           0.566           1.271           14.290 secs
06      0.542           0.552           1.190           17.491 secs, saving ...
07      0.536           0.547           1.241           14.598 secs, saving ...
08      0.530           0.539           1.200           15.505 secs, saving ...
09      0.524           0.534           1.271           14.136 secs, saving ...
10      0.520           0.535           1.447
```

- 最佳化cross entropy
- RMSE的分數與之前的Model相比是無意義的 ⇒ Label值域已經轉變!

Recommendation Engine with DNN

MF with DNN(Cross Entropy), 活用Embedding



找出跟Toy Story相似的電影
- ❏ Monsters, Inc
- ❏ Toy Story3
- ❏ Toy Story2

...

```
# cross entropy訓練出來的model
most_like(model_xent, 0, k=11)
```

```
INFO:tensorflow:Restoring parameters from ./model/model_mf_with_dnn_xent\model-11
```

	movieId	title	genres
0	0	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
3805	3805	Monsters, Inc. (2001)	Adventure|Animation|Children|Comedy|Fantasy
7556	7556	Toy Story 3 (2010)	Adventure|Animation|Children|Comedy|Fantasy|IMAX
7373	7373	Black Dynamite (2009)	Action|Comedy
7343	7343	Haunted World of El Superbeasto, The (2009)	Action|Animation|Comedy|Horror|Thriller
3419	3419	Shrek (2001)	Adventure|Animation|Children|Comedy|Fantasy|Ro...
215	215	Gordy (1995)	Children|Comedy|Fantasy
7653	7653	Red (2010)	Action|Comedy
7525	7525	Exit Through the Gift Shop (2010)	Comedy|Documentary
2506	2506	Toy Story 2 (1999)	Adventure|Animation|Children|Comedy|Fantasy
1656	1656	Sleeping Beauty (1959)	Animation|Children|Musical

MF with DNN - Metrics and Comparison

Model	MF	MF with history	MF with DNN(MSE)	MF with DNN (Cross Entropy)
RMSE Loss	0.92	0.92	**0.82**	~~1.44~~ **cross entropy loss: 0.53(無法與RMSE相比)**
ROC AUC	0.74	0.74	0.80	**0.81**
NDCG	strict: 0.63 normal: 0.76	strict: 0.63 normal: 0.76	**strict: 0.73** **normal: 0.83**	strict: 0.72 normal: 0.82
Precision at 10	strict: 0.50 normal: 0.63	strict: 0.51 normal: 0.63	**strict: 0.59** **normal: 0.68**	**strict: 0.59** **normal: 0.68**

Lab: lab_reco_model_mf_dnn.ipynb (20 minutes)

搜尋 ”Do:” 就可以找到可能要修改的位置

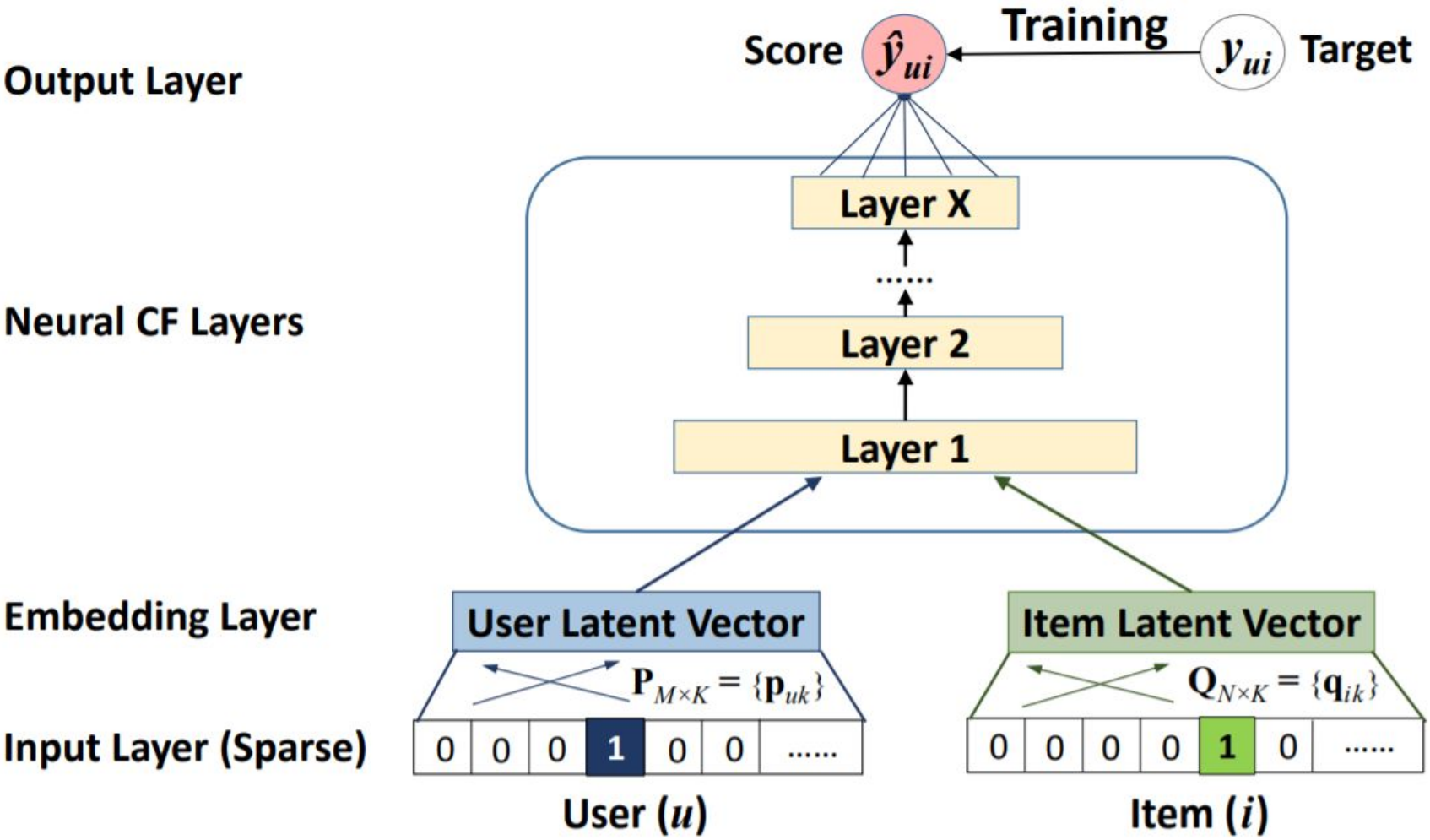
lab filename	lab_reco_model_mf_dnn.ipynb
target	您可能要修改的參數: ❑ learning rate ❑ dim: dimension of latent factor(user and movie) ❑ dropout ❑ loss function ❑ optimizer ❑ DNN structure
ideal score	valid rmse loss: 降低到0.82左右 roc auc: 0.80左右 ndcg: strict condition 0.60左右

Model MF with DNN 彈性的應用

- 對於新的user
  - 無使用紀錄
    - 推薦最受歡迎商品
  - 一點點使用紀錄
    - Content base recommendation(trained item embedding)
  - 大量使用紀錄
    - Model recommendation
- 對於新進來的電影
  - 將Metadata帶入model取得embedding就可以使用
    - ex: genres、avg_rating、year、director、actors(actresses)...
  - 不使用指向性的Feature ⇒ Ex: Movie ID
- 如果真的沒有負向資訊
  - Negative sampling for each user

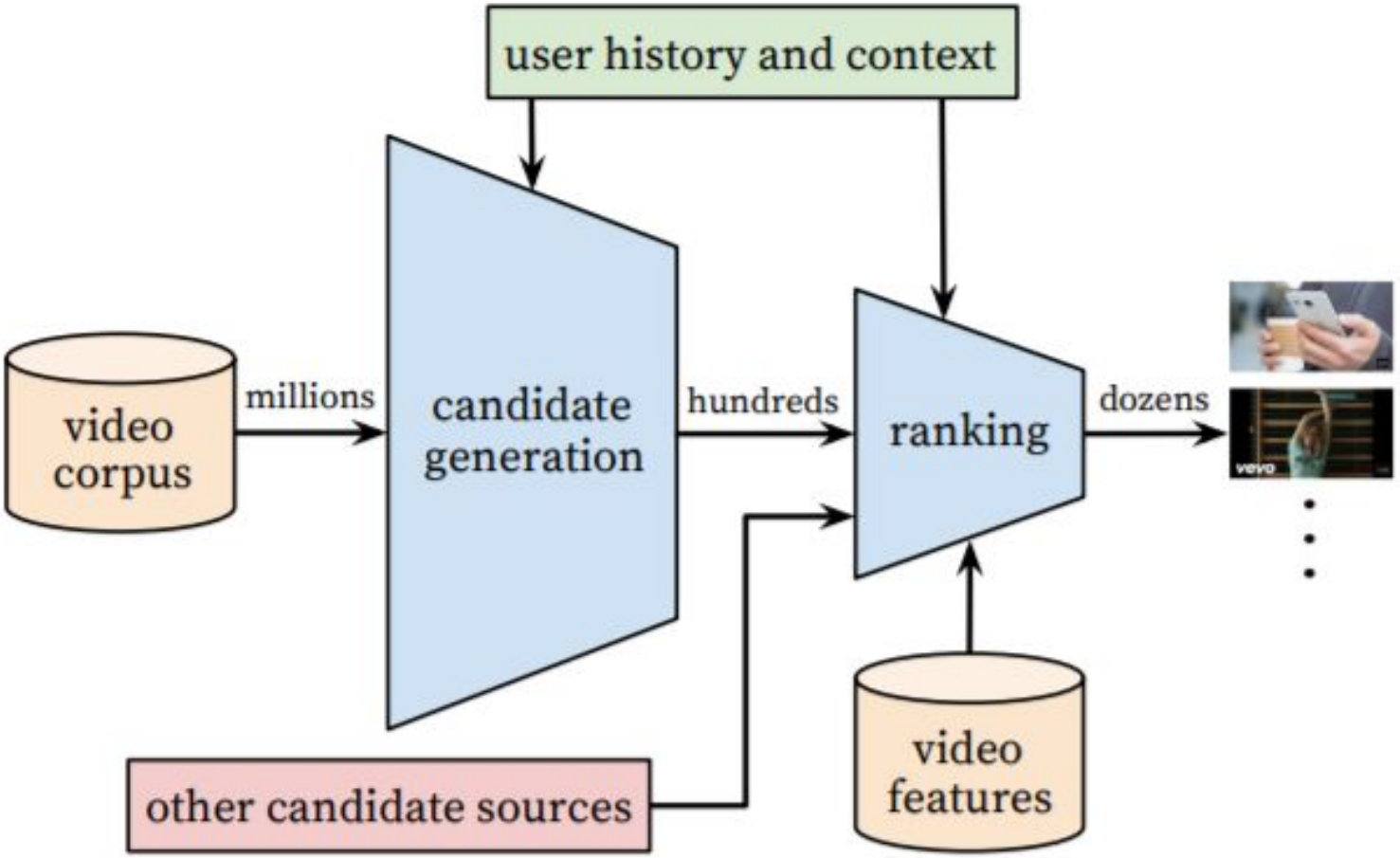
Appendix:
Other Recommendations Methods

Neural Collaborative Filtering



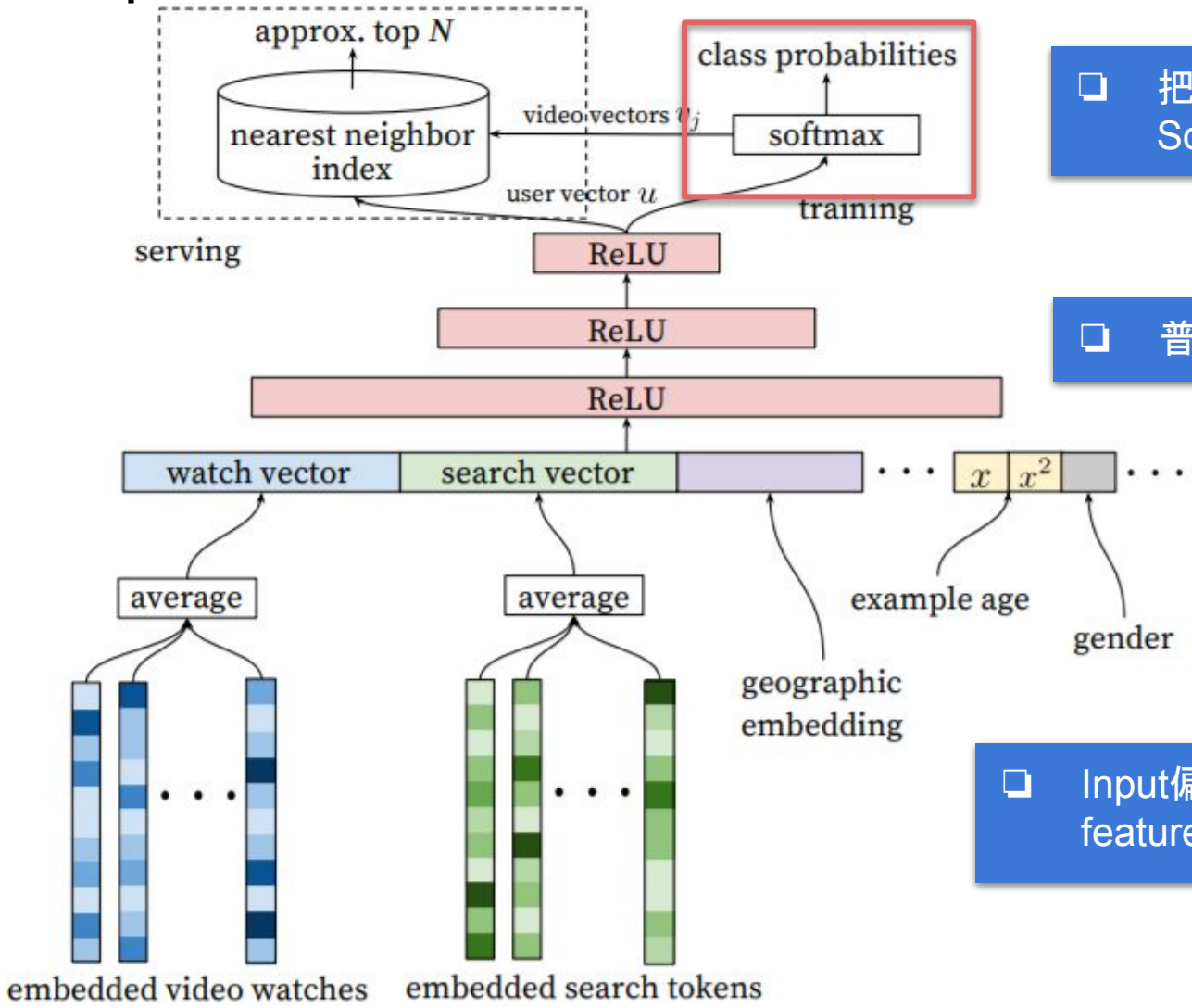
Paper: https://arxiv.org/pdf/1708.05031.pdf

Deep Neural Networks for YouTube Recommendations



Paper: [Google youtube recommendation]

Deep Neural Networks for YouTube Recommendations

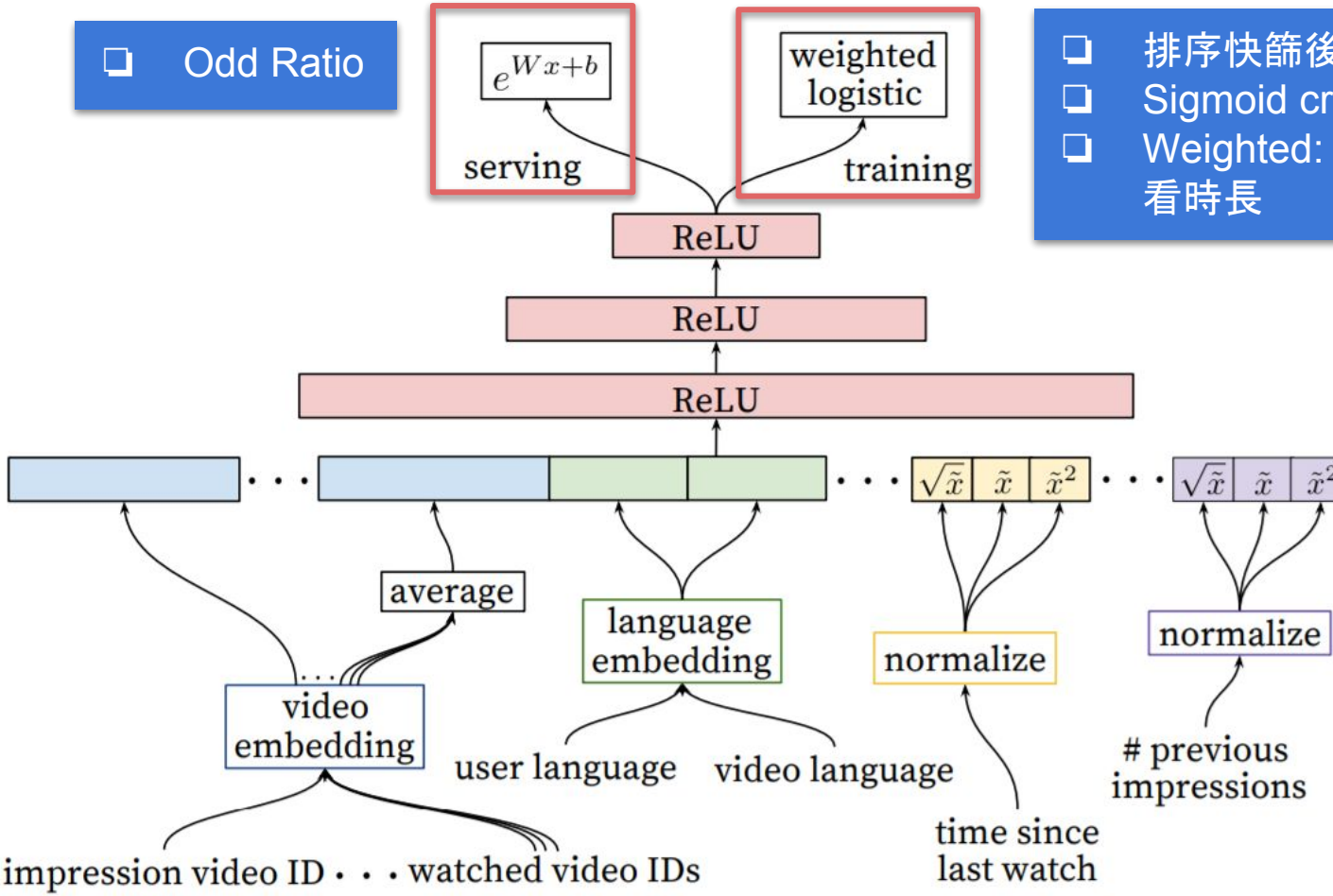


- 把所有Video當作類別 ⇒ Softmax cross entropy
- 普通DNN架構 + ReLU
- Input偏向user side的featuress(包含互動features)

Paper: [Google youtube recommendation]

Deep Neural Networks for YouTube Recommendations

- Odd Ratio



- 排序快篩後的Videos
- Sigmoid cross entropy
- Weighted: user對video的觀看時長

- Input偏向item side features

Paper: [Google youtube recommendation]

Conclusion

- 不建議一招打天下
  - 以Model推薦為主，Content base recommendation為輔
  - Hybrid Recommendation，各取所長

  ⇒ 選擇CP值高的做法

- 評估你的系統
  - KKBox like system ⇒ 考慮以Content base為主(CNN、RNN)
  - NLP like system(處理文章、文字)，e.g: 線上電子書
    - 考慮以Content base為主(RNN)
    - 以關鍵字關聯性做推薦 ⇒ #Hashtag…

Conclusion

- Matrix Factorization (Collaborative Filtering)
  - 近代最多人使用的方式
  - 最容易加入Context Information，可融合CNN、RNN

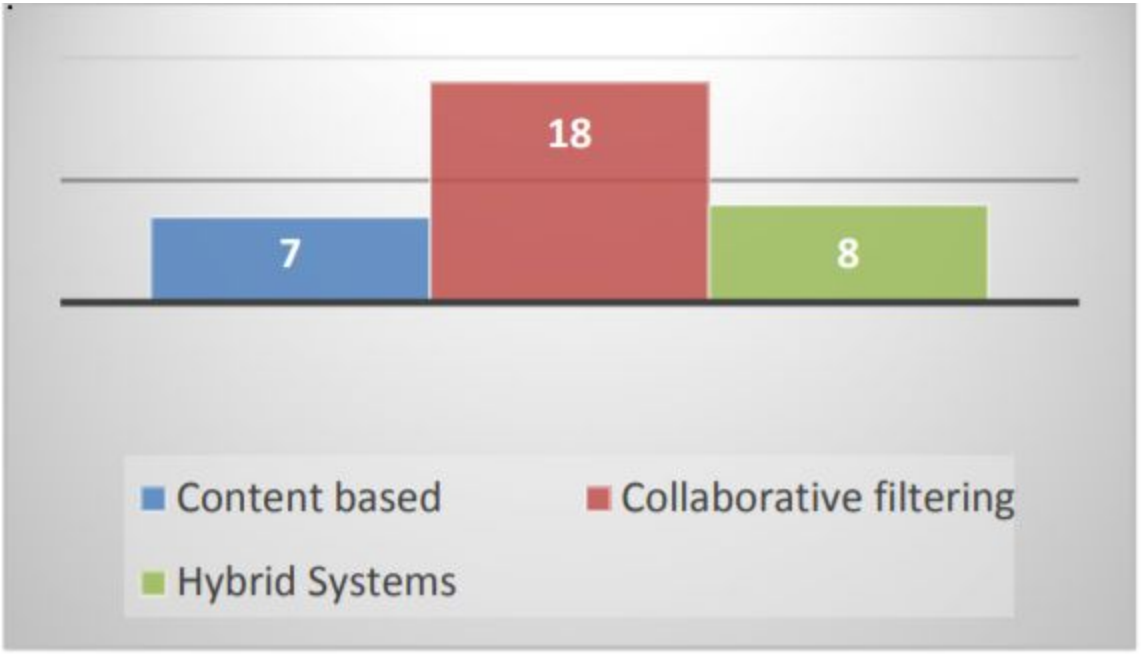


Fig.1: Distribution of publication counts in three recommender system categories.

Paper: [Use of Deep Learning in Modern Recommendation System: A Summary of Recent Works]

Reference

- 台大電機李宏毅教授Youtube頻道
- 手把手深度學習實務
- Paper: Google Youtube Recommendation
- Paper: Neural Collaborative Filtering
- Paper: Modern Recommendation System

- 若有課程上的建議，歡迎來信
  gary.chen@mile.cloud and johann.chu@mile.cloud