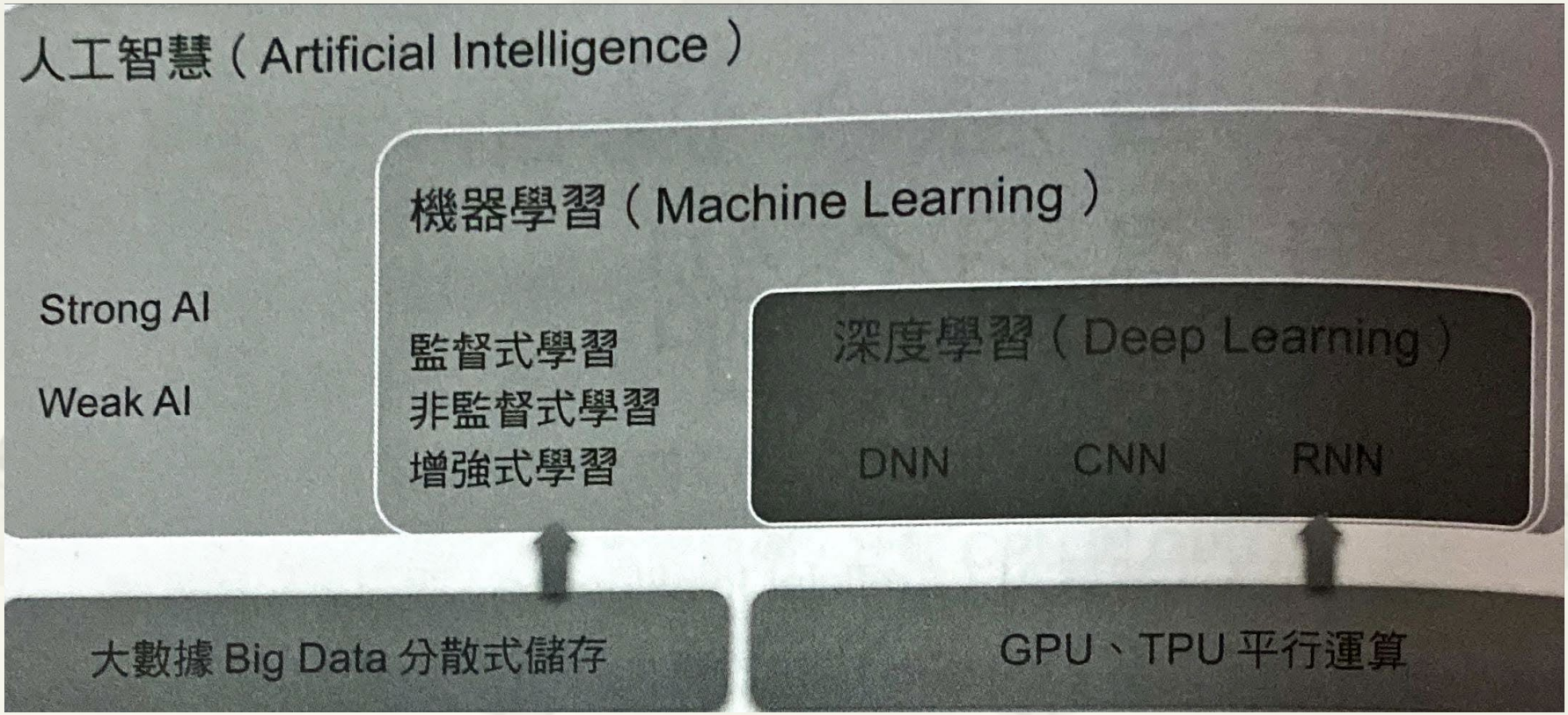
Chapter1

人工智慧、機器學習、深度學習介紹

關係圖


人工智慧（Artificial Intelligence）
機器學習（Machine Learning）
Strong AI
Weak AI
監督式學習
非監督式學習
增強式學習
深度學習（Deep Learning）
DNN
CNN
RNN
大數據 Big Data 分散式儲存
GPU、TPU 平行運算

人工智慧

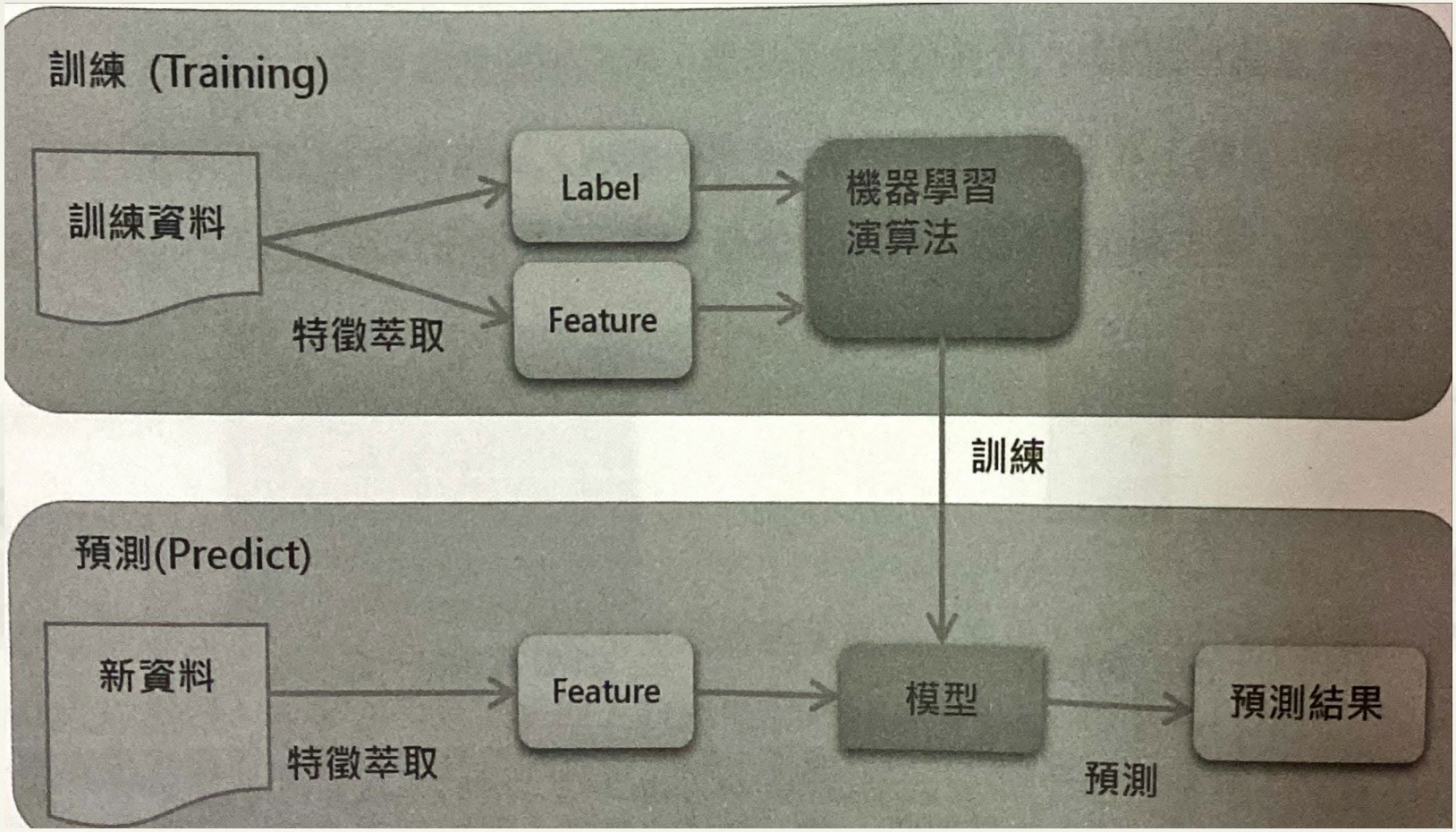
- **強人工智慧(Wrong AI)：**

機器能具有與人類相同完整的認知能力。

- **弱人工智慧(Weak AI)：**

機器不需要具有與人類相同完整的認知能力，只要設計的看起來像具有智慧就好。

機器學習


訓練 (Training)
訓練資料
特徵萃取
Label
Feature
機器學習
演算法
訓練
預測(Predict)
新資料
特徵萃取
Feature
模型
預測
預測結果

機器學習

- **大數據big data分散式儲存。**

- **監督式學習：**具備特徵與預測目標，透過演算法訓練並建立模型。

  二元分類：預測目標兩種，是非題。

  多元分類：預測目標多種，選擇題。

  回歸分析：預測目標連續值，計算題。

機器學習

- **非監督式學習**：具備特徵，沒有預測目標。

  分群：Cluster集群演算法，將資料分成幾個相異性最大的群組，而群組內相似程度最高。

- **強化式學習**：藉由定義（動作、狀態、獎勵等），不斷訓練機器循序漸進。

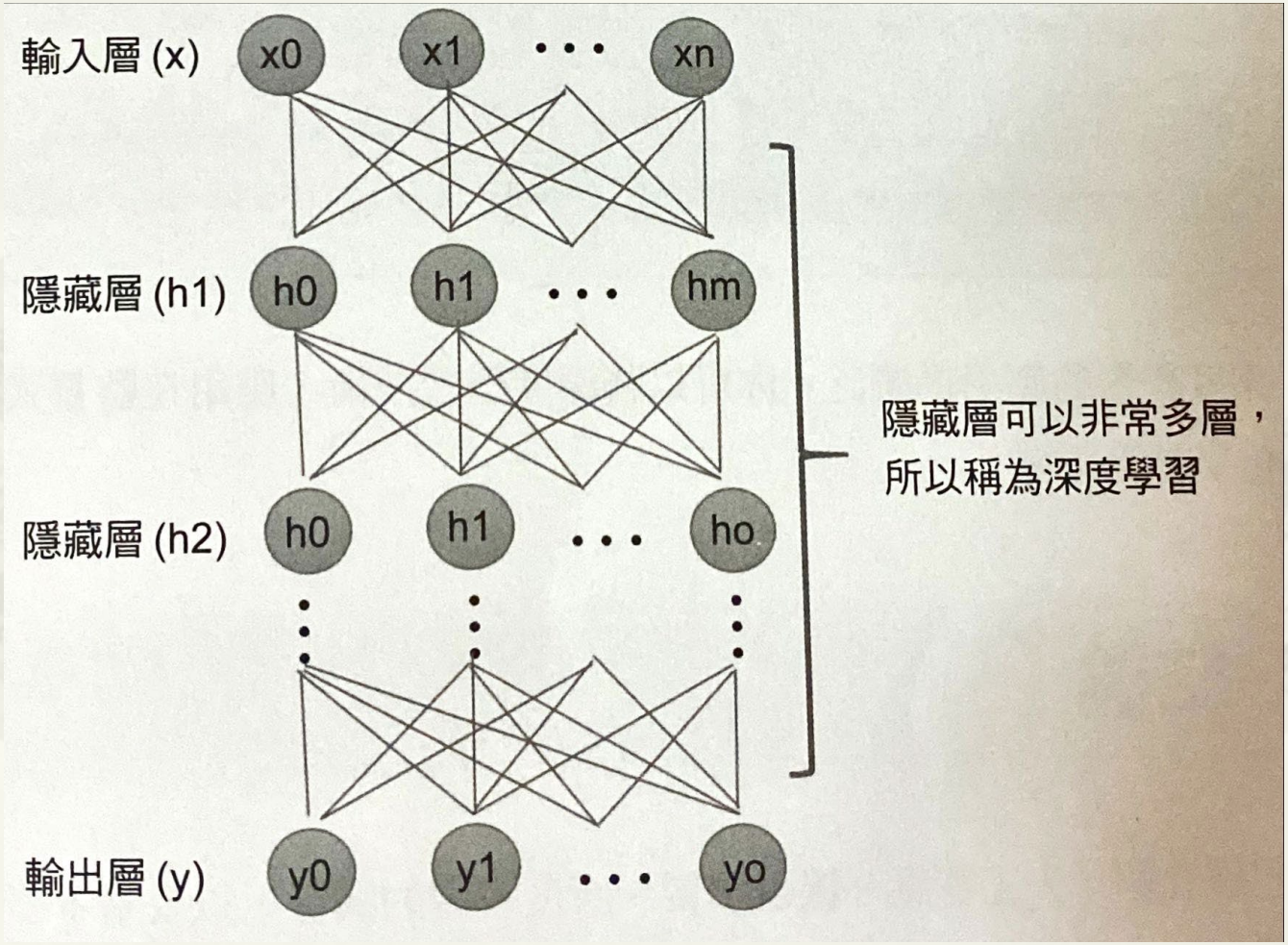
  Q-learing、TD(Temporal Difference)、Sarsa

深度學習

- GPU、TPU平行運算。

- 方便電腦模擬，將神經元分成多層次，模擬神經網路，通常有1個輸入層、1個輸出層、多個隱藏層。

深度學習


輸入層 (x)
x0
x1
xn
隱藏層 (h1)
h0
h1
hm
隱藏層 (h2)
h0
h1
ho
輸出層 (y)
y0
y1
yo
隱藏層可以非常多層，
所以稱為深度學習

深度學習

- **多層感知器：**

一種人工神經網路的基本形式。由多個神經元組成，通常包含一個或多個隱藏層，以及一個輸入層和一個輸出層。

- **深度神經網路DNN：**

多層感知器的一種擴展，通常包含更多的隱藏層和神經元，學習並表示更複雜的關係。

深度學習

- **卷積神經網路CNN：**

應用在影像辨識、影像處理和自然語言處理等，設計靈感來自於人類視覺系統。

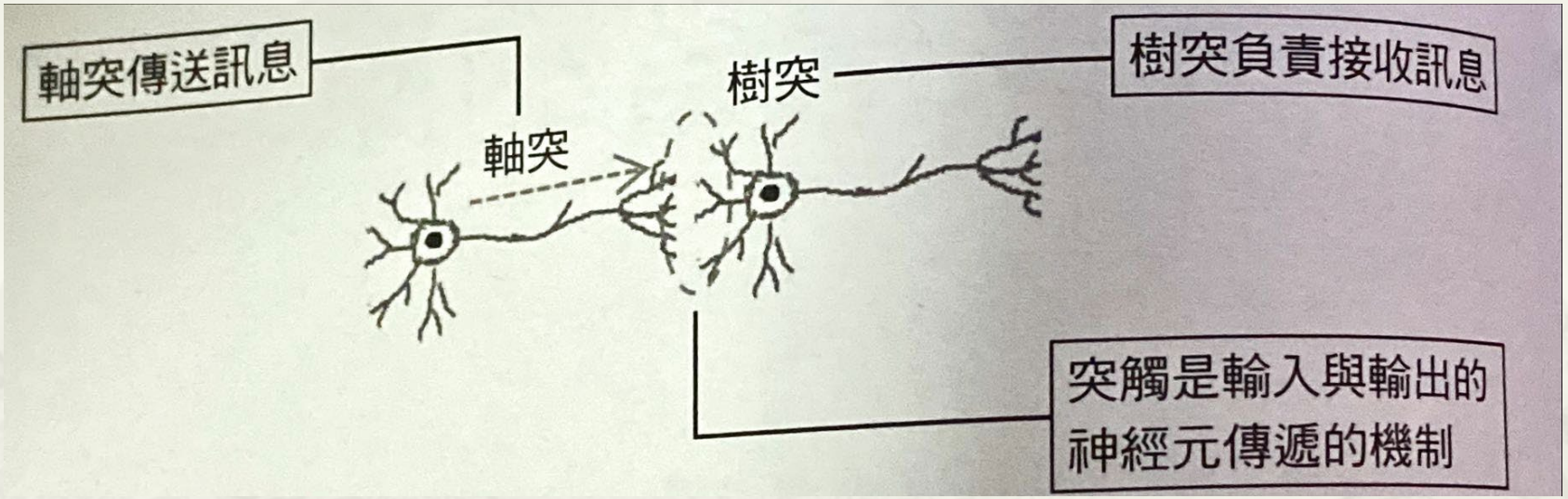
- **遞迴神經網路RNN：**

具有回饋循環，適用於處理序列數據，能夠記住先前的信息，並在後續時間使用這些信息來影響輸出，如文字、語音、時間序列等。

Chapter2
深度學習的原理

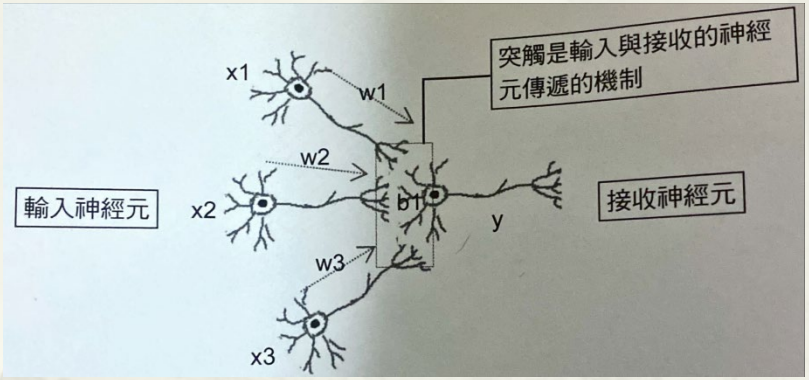
深度學習

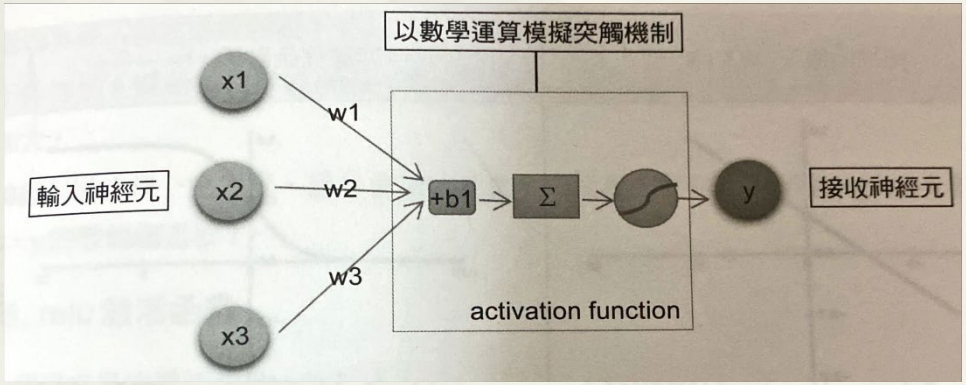
- 神經元的訊息傳導介紹



深度學習

- 模擬神經元訊息傳導





- 用數學公式表示

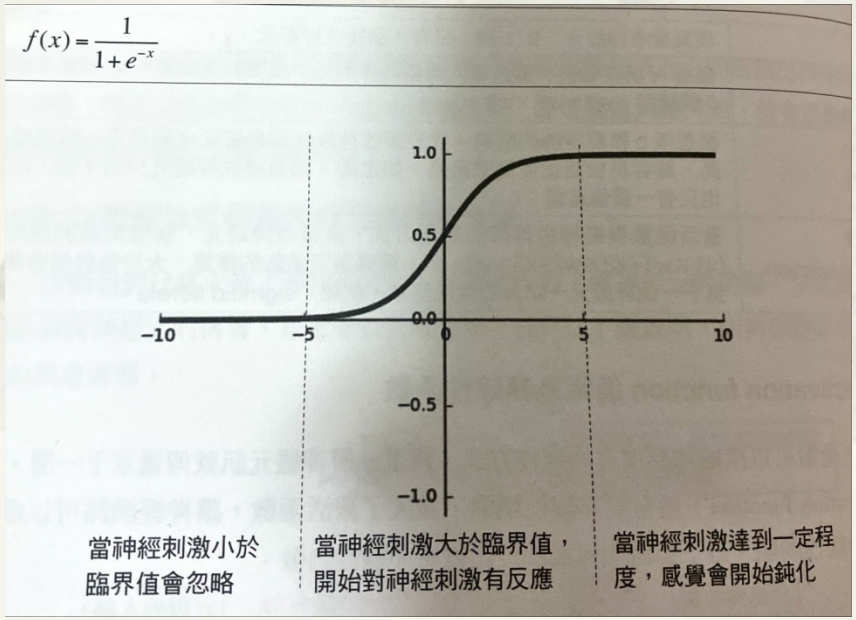
$$y=\text{activation function}(x1 \times w1+x2 \times w2+x3 \times w3+b1)$$

X：輸入
Y：接收
W：權重
B：偏差值
Activation function：激活函數

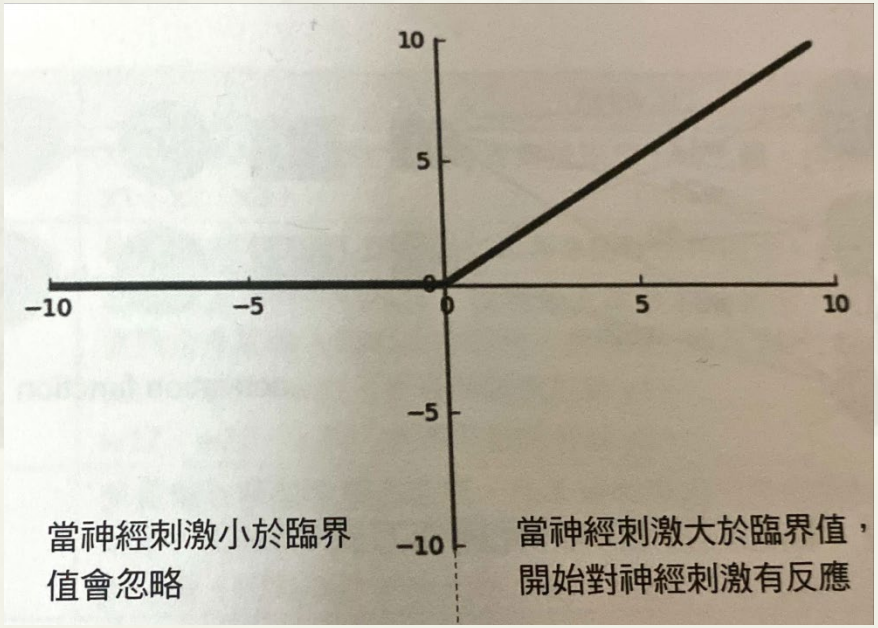
深度學習

- Activation function通常為非線性函數

Sigmoid激活函數

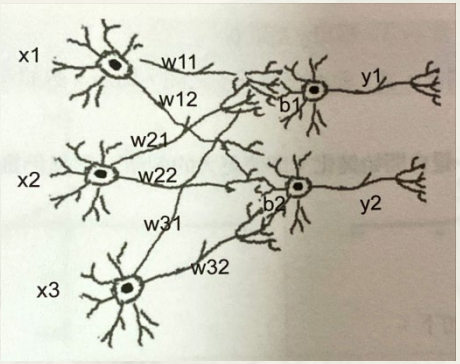


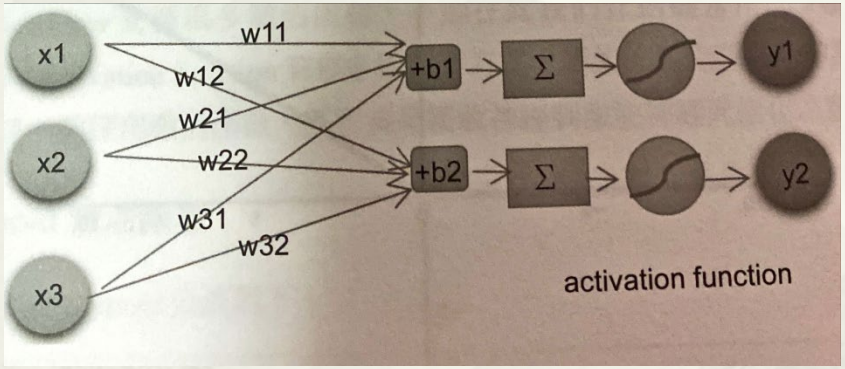
Relu激活函數



深度學習

- 以矩陣運算模擬神經網路





- 用數學公式表示

y1=activation function(x1×w11+x2×w21+x3×w31+b1)

y2=activation function(x1×w12+x2×w22+x3×w32+b2)

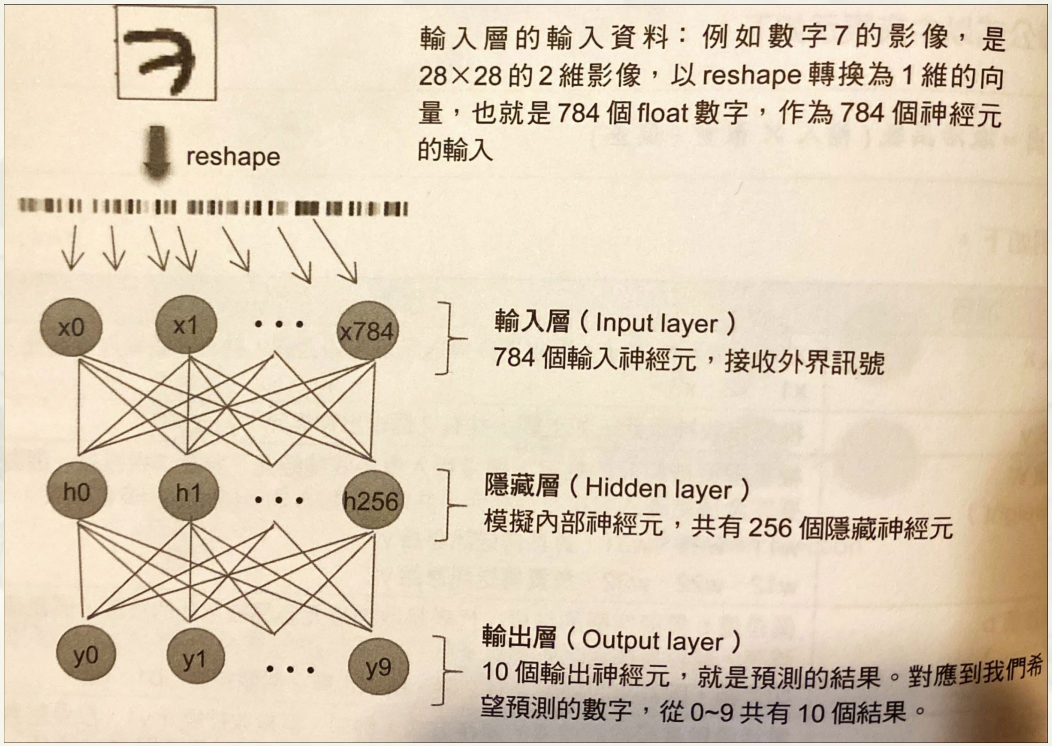
合併

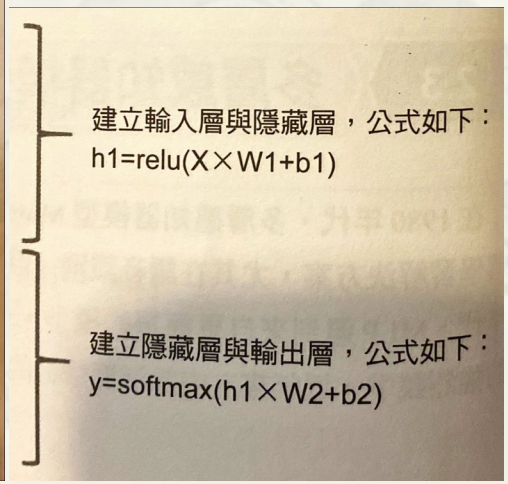
$$[y1 \quad y2] = activation\left([x1 \quad x2 \quad x3] \times \begin{bmatrix} w11 & w12 \\ w21 & w22 \\ w31 & w32 \end{bmatrix}\right) + [b1 \quad b2]$$

X：輸入　Y：接收　W：權重　B：偏差值　Activation function：激活函數

深度學習

- 多層感知器辨識Mnist手寫數字影像



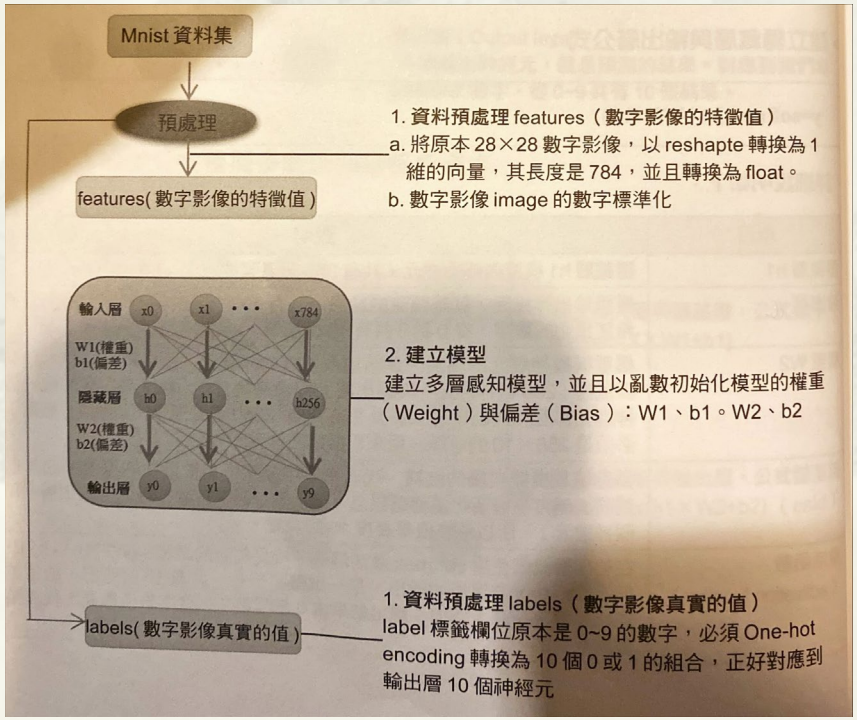


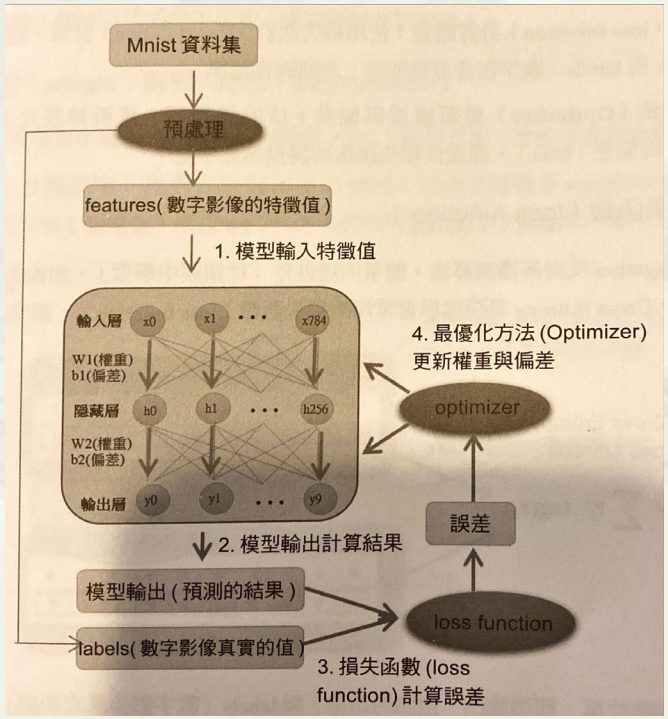
深度學習

- Sigmoid、Relu、Softmax函數應用
- Sigmoid激活函數應用：二元分類。
- Relu激活函數應用：卷積神經網路CNN。
- Softmax激活函數應用：多元分類。

深度學習

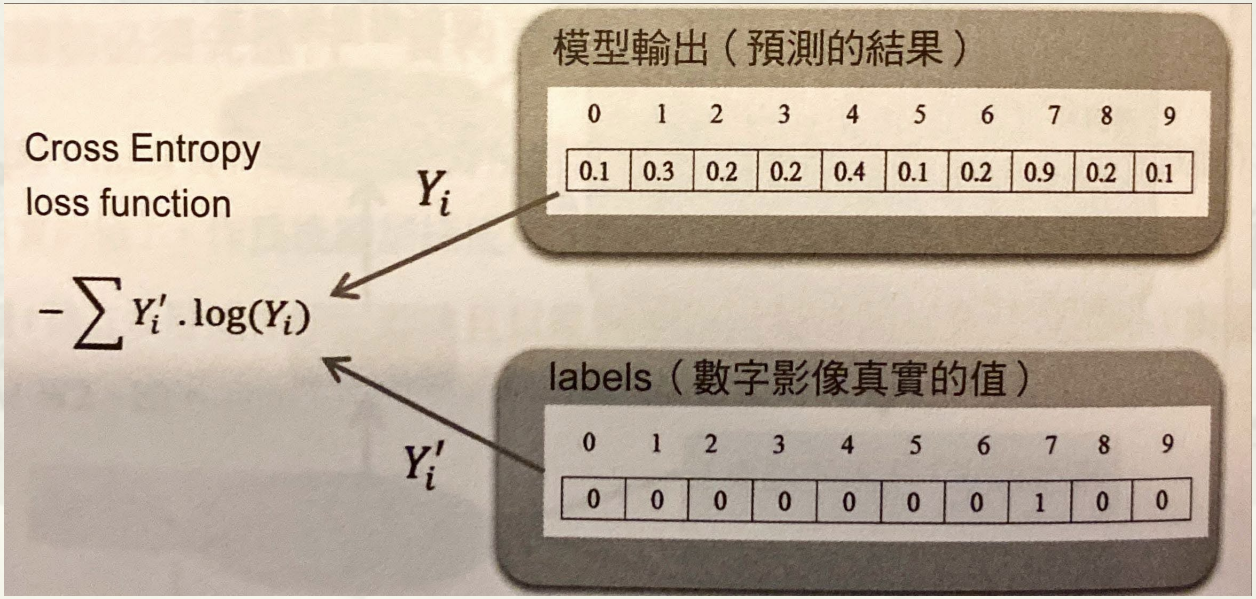
- Back Propagation反向傳播演算法
- 監督式學習，必須輸入features、labels。





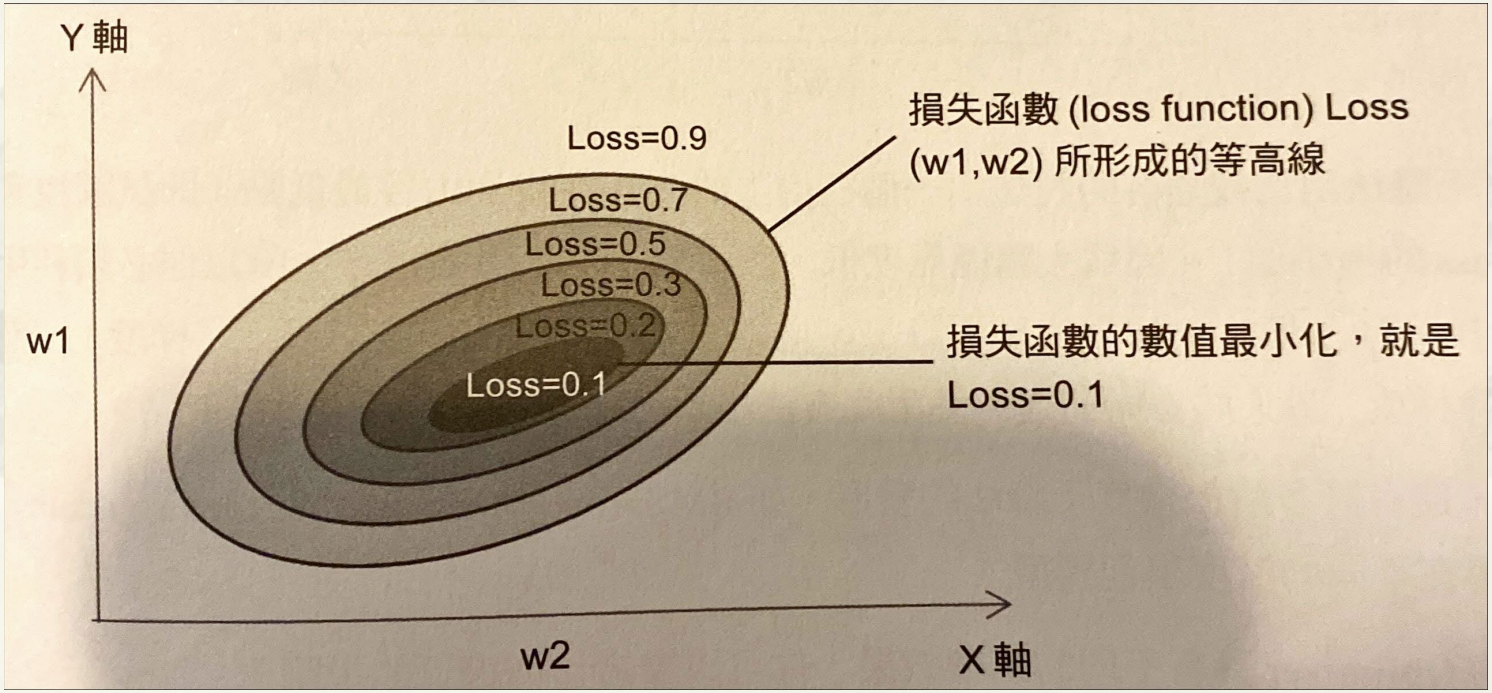
深度學習

- 損失函數
- 從錯誤中學習，計算誤差。



深度學習

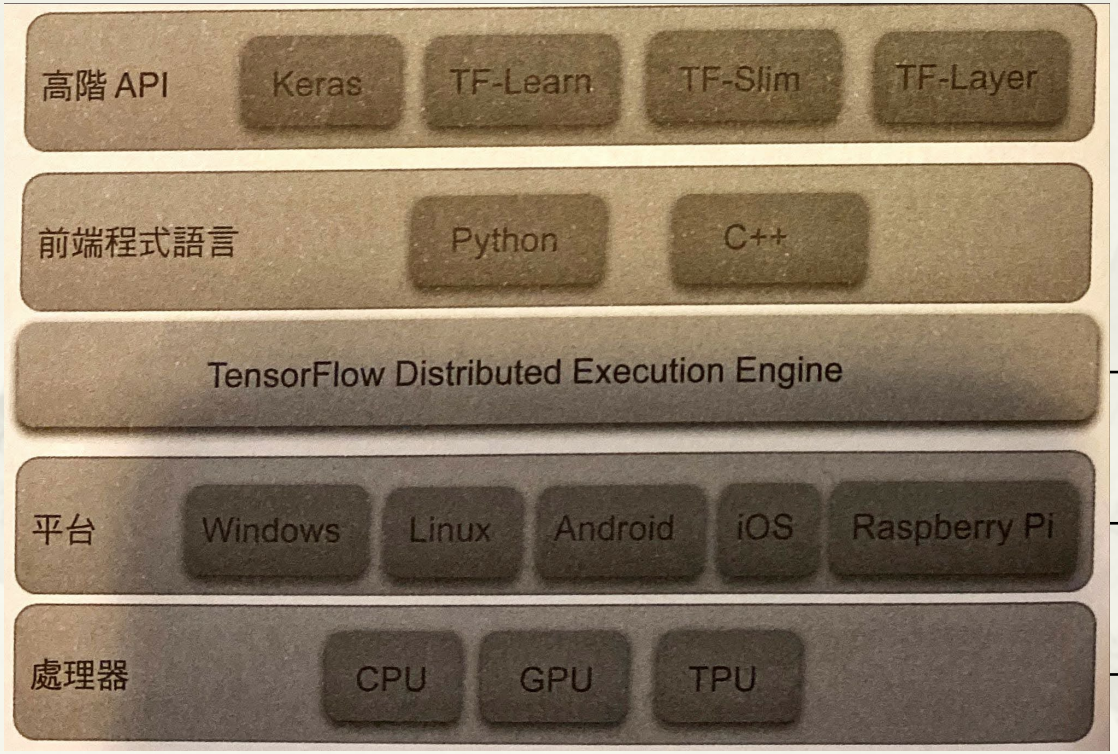
- 最佳化方法：SGD梯度下降法。
- 二維權重與損失函數



Chapter3
TensorFlow與Keras介紹

TensorFlow

- TensorFlow架構



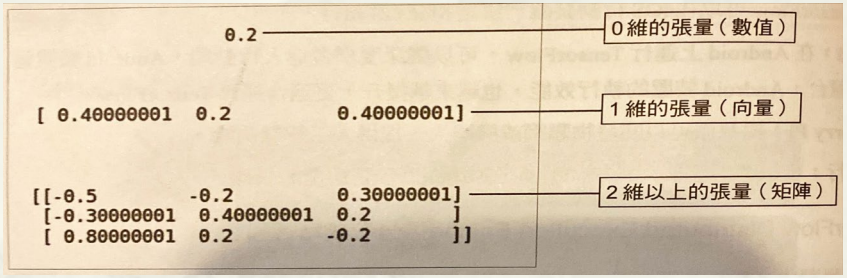
分散式執行引擎，可同時在數台機器上訓練模型。

跨平台能力。

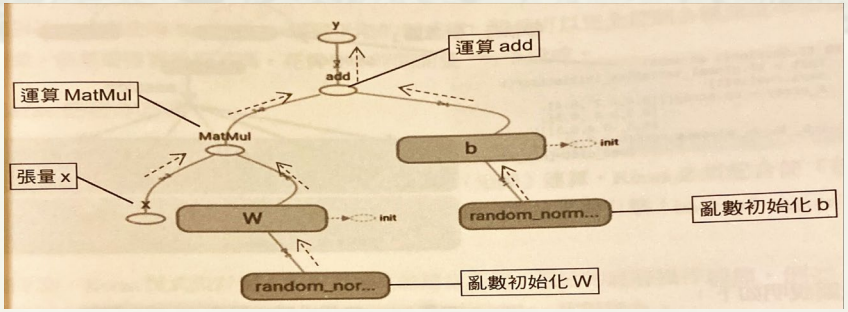
可雲端執行。

TensorFlow

- Tensor張量



- Flow資料流程：使用TensorFlow模型建立計算圖，就可在不同平台上執行計算圖。



算式：y=MatMul(x,W)+b

TensorFlow

- TensorFlow建立與執行計算圖

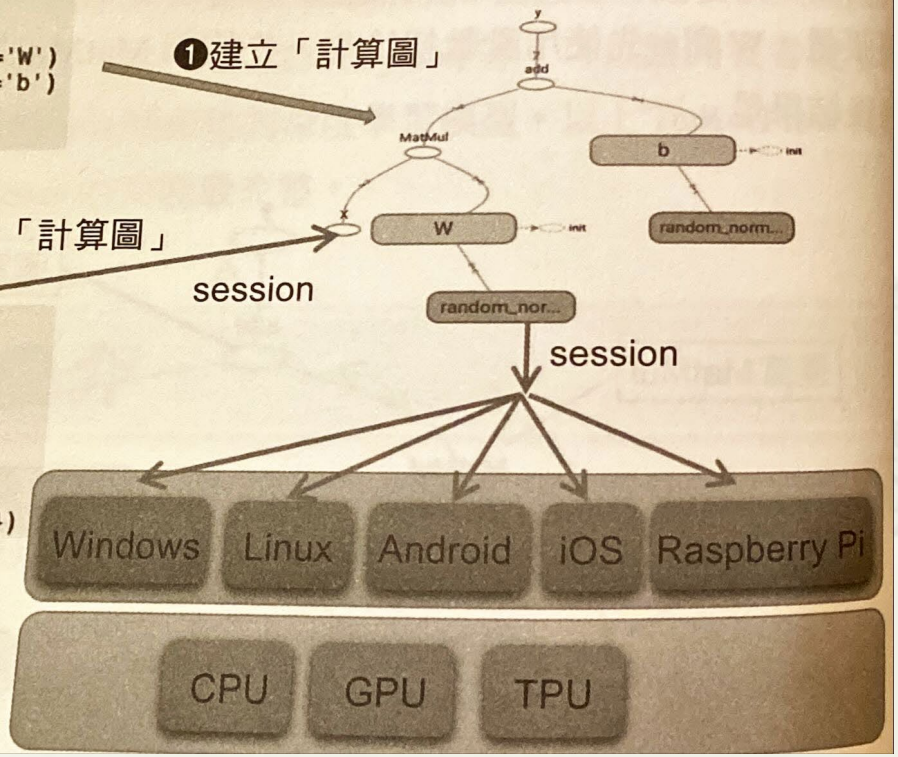
```
import tensorflow as tf
import numpy as np
W = tf.Variable(tf.random_normal([3, 2]),name='W')
b = tf.Variable(tf.random_normal([1, 2]),name='b')
X = tf.placeholder("float", [None,3],name='X')
y=tf.nn.sigmoid(tf.matmul(X,W)+b,'y')
```

❶建立「計算圖」

❷執行「計算圖」

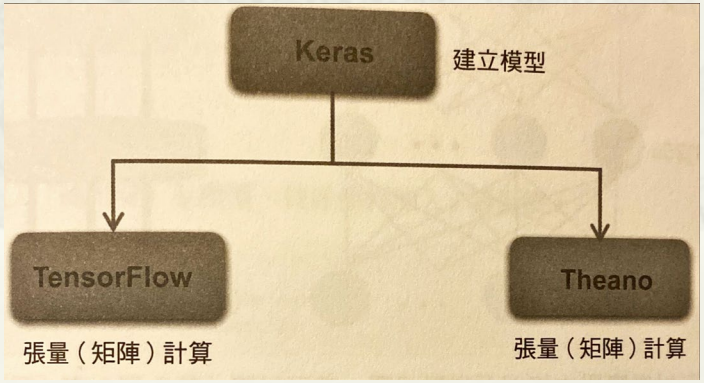
session

```
with tf.Session() as sess:
    init = tf.global_variables_initializer()
    sess.run(init)
    X_array = np.array([[0.4,0.2 ,0.4],
                        [0.3,0.4 ,0.5],
                        [0.3,-0.4,0.5]])
    (_b,_W,_X,_y)=sess.run((b,W,X,y),
                           feed_dict={X:X_array})
```



Keras

- 高階深度學習程式庫。
- 只處理模型建立、訓練、預測等。
- 後端引擎：Theano、TensorFlow。

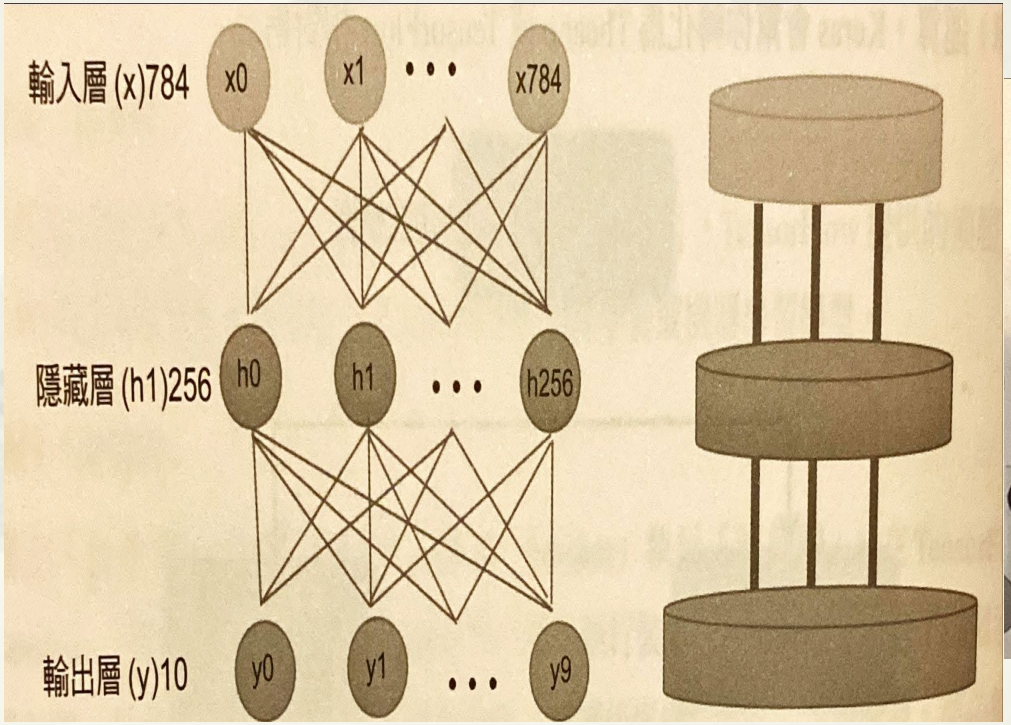


Keras特色

- 簡單快速。
- 內建各式類神經網路層級：CNN、RNN。
- 透過後端引擎可在CPU、TPU運行。
- Keras開發的程式碼更簡潔、可讀性高、易維護、生產力高，說明文件淺顯易懂。

Keras

- 程式設計模式



加入「輸入層」與「隱藏層」

```
(Dense(units=256,
       input_dim=784,
       kernel_initializer='normal',
       activation='relu'))
```

加入「輸出層」

```
(Dense(units=10,
       kernel_initializer='normal',
       activation='softmax'))
```

Keras

- **程式設計模式**

- 建立Sequential模型（蛋糕架）

```
model = Sequential( )
```

- 加入「輸入層」與「隱藏層」至模型（第一層蛋糕）

```
model.add(Dense(units=256,
                input_dim=784,
                kernel_initializer='normal' ,
                activation='relu' ))
```

- 加入「輸出層」至模型（第二層蛋糕）

```
model.add(Dense(units=10,
                kernel_initializer='normal',
                activation= 'softmax'))
```

Keras與TensorFlow比較

	Keras	TensorFlow
學習難易度	簡單	困難
使用彈性	中等	高
開發生產力	高	中等
執行效能	高	高
適合使用者	初學者	進階使用者
張量(矩陣)運算	不需自行設計	需自行設計