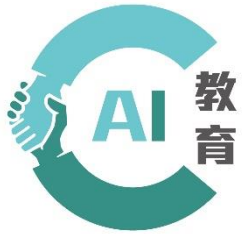
NARLabs 財團法人國家實驗研究院
國家高速網路與計算中心
National Center for High-performance Computing



Keras 及 TensorFlow 深度學習介紹

莊家雋 助理研究員

國家高速網路與計算中心

課程大綱

- Python 基礎
  - Python、Numpy、Matplot、Panda
- 深度學習原理
  - Neural network 基礎
  - Keras v.s. Tensorflow
- Keras處理手寫數字辨識資料集介
  - 建立Multiple Layer perceptron model (MLP)
  - 建立 convolutional neural network (CNN)
- Tensorflow手寫數字辨識資料集介
  - 建立Multiple Layer perceptron model (MLP)
  - 建立 convolutional neural network (CNN)

Jupyter Notebook

- **Cell：**
  - **Code ＆Markdown**
- 運行一個cell，就會產生結果
  - Shift + enter
- **Online document**
  - **Shift + tab**

```
In [2]: fashion_mnist = keras.datasets.fashion_mnist

        (train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data()
```

```
Type:            ndarray
String form:     [9 0 0 ... 3 0 5]
Length:          60000
File:            /usr/local/lib/python2.7/dist-packages/numpy/__init__.py
Docstring:       <no docstring>
Class docstring:
ndarray(shape, dtype=float, buffer=None, offset=0,
        strides=None, order=None)

An array object represents a multidimensional, homogeneous array
of fixed-size items.  An associated data-type object describes the
```

s to learn.

n array of integers, ranging from 0 to 9. These correspond

1 Trouser

Python 基礎 - Variable and Type

```
In [1]: b = '456'
        type(b)
```

```
Out[1]: str
```

```
In [3]: c = 8.70
        type(c)
```

```
Out[3]: float
```

數值可以做運算：

```
In [6]: a = 5
        b = 8
        print (a+b)
        print (max(a,b))
```

```
13
8
```

不同的形態運算要注意，不要把字串和數值加在一起、否則會產生TypeError。

```
In [8]: a = 5
        b = '8'
        print (a+b)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-8-3bbcac7fa58d> in <module>()
      1 a = 5
      2 b = '8'
----> 3 print (a+b)

TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

可以用 int(), str(), float() 來轉換變數型態

```
In [9]: a = 5
        b = int('8')
        print (a+b)
```

```
13
```

Python 基礎 – if-else

```
if condition:
    do-something()
else:
    do-something-else()
```

```
if condition1:
    do-something()
elif condition2:
    do-something-2()
else:
    do-something-else()
```

Python 基礎 – for loop

```
for i in range(n):
    do-something(i)
```

```
for i in list:
    do-something(i)
```

Python 基礎 – list

```
xs = [3, 1, 2]   # Create a list
print xs, xs[2]
print xs[-1]     # Negative indices count from the end of the list; prints "2"
```

```
[3, 1, 2] 2
2
```

```
xs[2] = 'foo'    # Lists can contain elements of different types
print xs
```

```
[3, 1, 'foo']
```

```
xs.append('bar') # Add a new element to the end of the list
print xs
```

```
[3, 1, 'foo', 'bar']
```

```
x = xs.pop()     # Remove and return the last element of the list
print x, xs
```

```
bar [3, 1, 'foo']
```

Python 基礎 – slicing

```
nums = range(5)     # range is a built-in function that creates a list of integers
print nums          # Prints "[0, 1, 2, 3, 4]"
print nums[2:4]     # Get a slice from index 2 to 4 (exclusive); prints "[2, 3]"
print nums[2:]      # Get a slice from index 2 to the end; prints "[2, 3, 4]"
print nums[:2]      # Get a slice from the start to index 2 (exclusive); prints "[0, 1]"
print nums[:]       # Get a slice of the whole list; prints ["0, 1, 2, 3, 4]"
print nums[:-1]     # Slice indices can be negative; prints ["0, 1, 2, 3]"
nums[2:4] = [8, 9] # Assign a new sublist to a slice
print nums          # Prints "[0, 1, 8, 9, 4]"
```

Python 基礎 – Dictionary

```
d = {'cat': 'cute', 'dog': 'furry'}  # Create a new dictionary with some data
print d['cat']       # Get an entry from a dictionary; prints "cute"
print 'cat' in d     # Check if a dictionary has a given key; prints "True"
```

```
cute
True
```

```
d['fish'] = 'wet'    # Set an entry in a dictionary
print d['fish']      # Prints "wet"
```

```
wet
```

```
print d['monkey']  # KeyError: 'monkey' not a key of d
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-24-2b04a5bbc99e> in <module>()
----> 1 print d['monkey']  # KeyError: 'monkey' not a key of d

KeyError: 'monkey'
```

Python 基礎 – Tuple

A tuple is an (immutable) ordered list of values.

```
In [4]: t = (5, 6, 7)        # Create a tuple
        print type(t)
        print t
        print t[0], t[1], t[2]
```

```
<type 'tuple'>
(5, 6, 7)
5 6 7
```

Python 基礎 – Functions

```
def sign(x):
    if x > 0:
        return 'positive'
    elif x < 0:
        return 'negative'
    else:
        return 'zero'

for x in [-1, 0, 1]:
    print sign(x)
```

```
def hello(name, loud=False):
    if loud:
        print 'HELLO, %s' % name.upper()
    else:
        print 'Hello, %s!' % name

hello('Bob')
hello('Fred', loud=True)
```

Python 基礎 – import

語法1： `import [module]`

```
# Import 整個 `random` module
import random

# 使用 `random` module 底下的 `randint` function
print(random.randint(0, 5))
```

語法2： `from [module] import [name1, name2, ...]`

```
# 從 `random` module 裡 import 其中一個 function `randint`
from random import randint

# 不一樣的是，使用 `randint` 的時候就不需要先寫 `random` 了
print(randint(0, 5))
```

語法3： `import [module] as [new_name]`


```
# Import 整個 `random` module，
# 但這個名字可能跟其他地方有衝突，因此改名成 `rd`
import random as rd

# 使用 `rd` 這個名稱取代原本的 `random`
print(rd.randint(0, 5))
```

Numpy基礎 – Arrays

```
a = np.array([1, 2, 3])  # Create a rank 1 array
print type(a)
print a.shape
print a[0], a[1], a[2]
print a.dtype
a[0] = 5                 # Change an element of the array
print a
```

```
<type 'numpy.ndarray'>
(3,)
1 2 3
int64
[5 2 3]
```

```
b = np.array([[1,2,3],[4,5,6]])   # Create a rank 2 array
print type(b)
print b.shape
print b[0]
print b[0,0]
b[0][0]=5
print b
```

```
<type 'numpy.ndarray'>
(2, 3)
[1 2 3]
1
[[5 2 3]
 [4 5 6]]
```

```
c = np.array(1)
print type(c)
print c.shape
printc[0]
print c
```

```
<type 'numpy.ndarray'>
()

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-7-eb5e8df49b9f> in <module>()
      2 print type(c)
      3 print c.shape
----> 4 printc[0]
      5 print c

NameError: name 'printc' is not defined
```

Numpy基礎 – Reshape

Reshape ¶

In [4]:
```
b1 = b.reshape([6,])
print b1.shape
print b1
```

```
(6,)
[5 2 3 4 5 6]
```

In [5]:
```
b2 = b.reshape([6,1])
print b2.shape
print b2
```

```
(6, 1)
[[5]
 [2]
 [3]
 [4]
 [5]
 [6]]
```

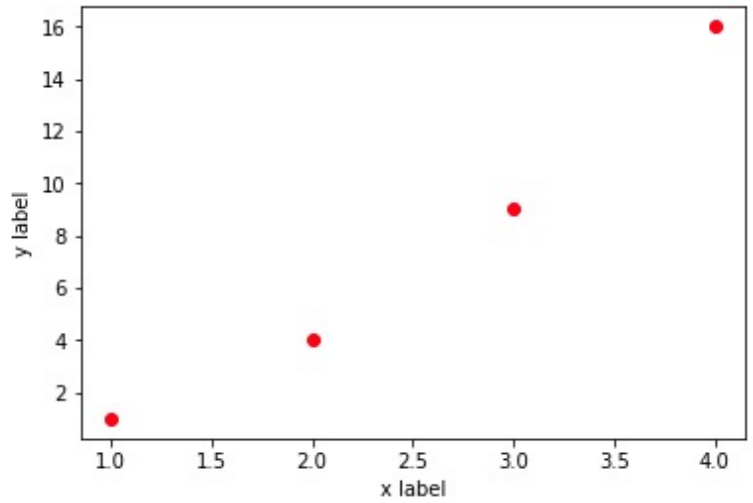
```
b3 = b.reshape([3,2,1])
print b3
```

```
[[[5]
  [2]]

 [[3]
  [4]]

 [[5]
  [6]]]
```

Matplot基礎 – plot

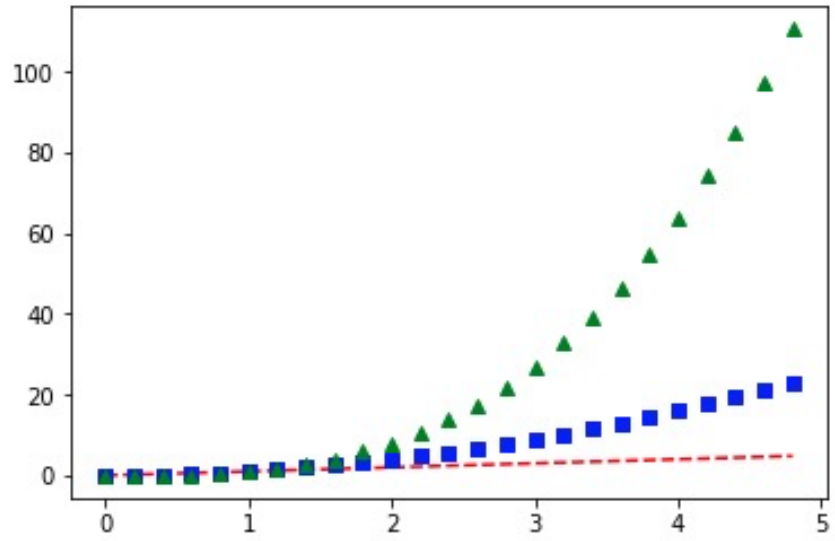
```
In [6]: plt.plot([1,2,3,4],[1,4,9,16],'ro')
        plt.ylabel('y label')
        plt.xlabel('x label')
```

```
Out[6]: Text(0.5,0,'x label')
```




```
In [12]: t = np.arange(0., 5., 0.2)
         plt.plot(t,t,'r--', t,t**2,'bs', t, t**3,'g^')
```

```
Out[12]: [<matplotlib.lines.Line2D at 0x7f5d31d6df50>,
          <matplotlib.lines.Line2D at 0x7f5d31d7b050>,
          <matplotlib.lines.Line2D at 0x7f5d31d7b850>]
```



Panda

- Numpy強化 array
- Panda強化 Discionary
  - Series: 時間序列資料
  - DataFrame: 二維結構化資料

```
In [29]: import pandas as pd # 引用套件並縮寫為 pd

groups = ["Movies", "Sports", "Coding", "Fishing", "Dancing", "cooking"]
num = [46, 8, 12, 12, 6, 58]

df = pd.DataFrame({"groups": groups,  "num": num})

df[:]
```

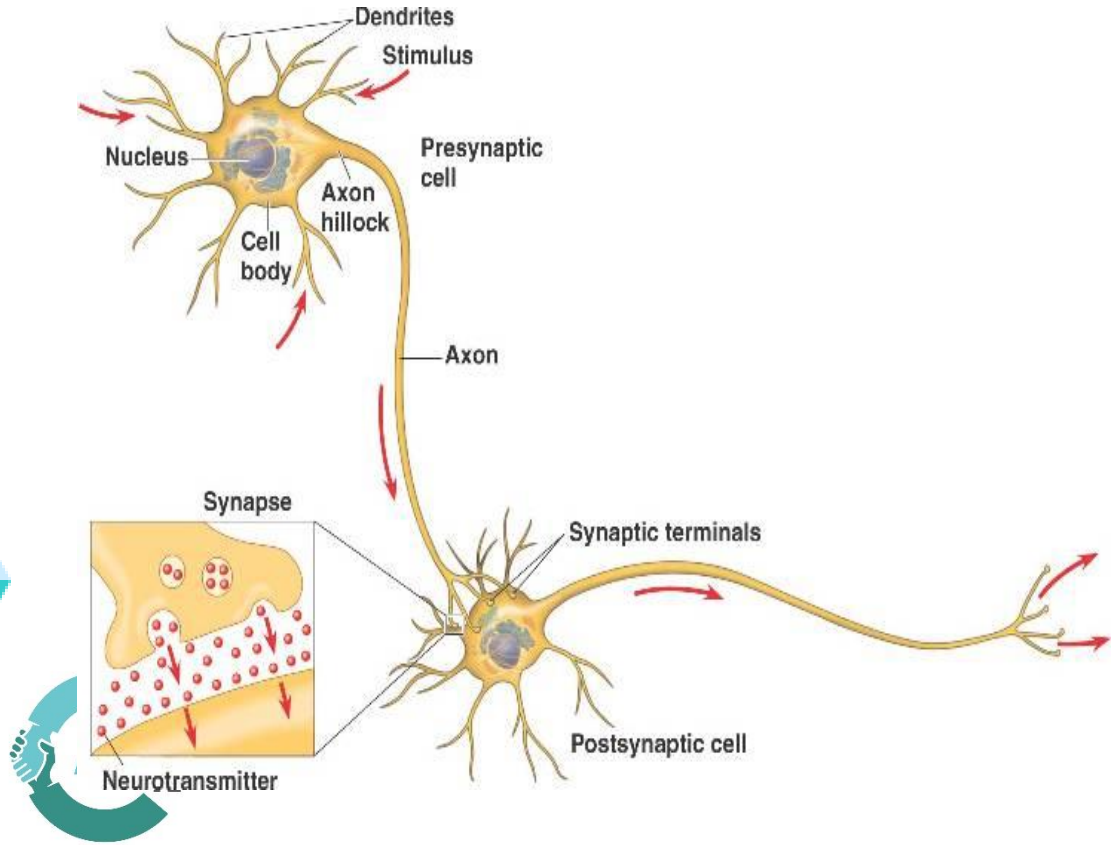
Out[29]:

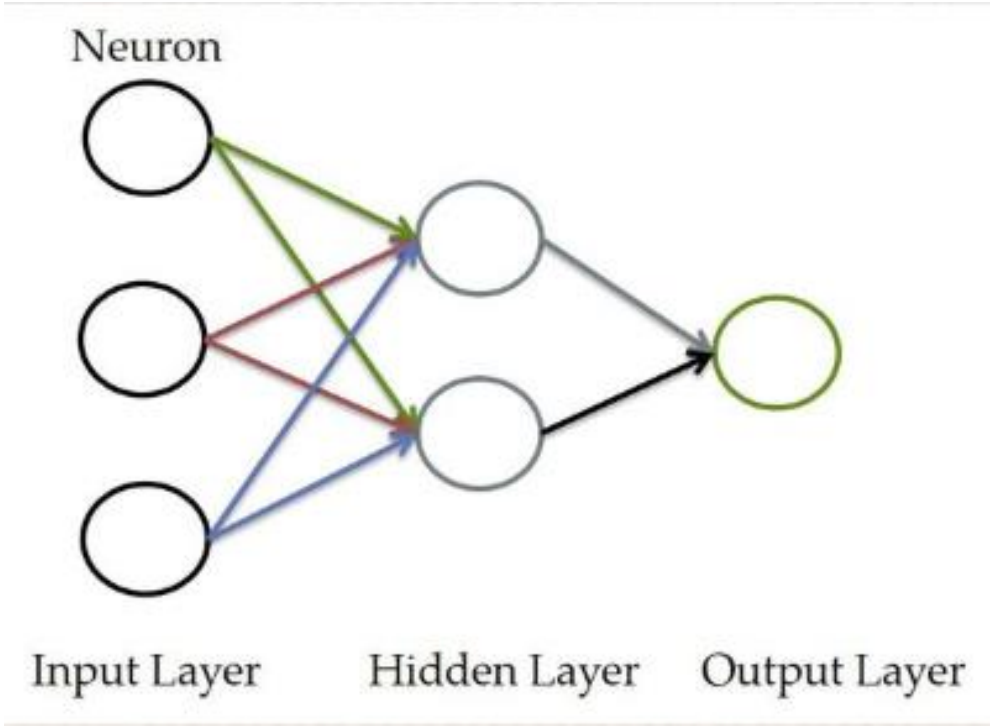
	groups	num
0	Movies	46
1	Sports	8
2	Coding	12
3	Fishing	12
4	Dancing	6
5	cooking	58

課程大綱

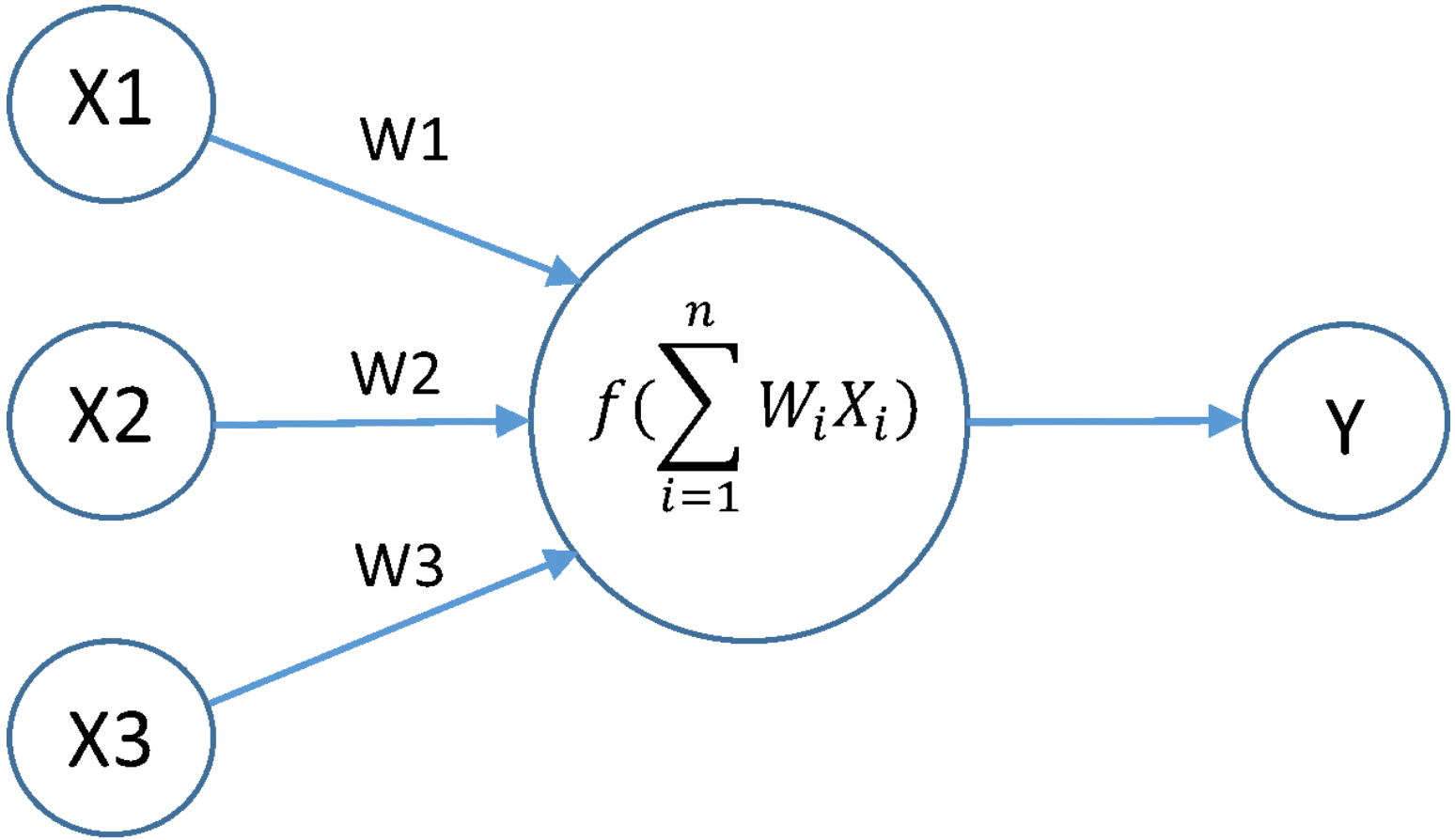
- Python 基礎
  - Python、Numpy、Matplot、Panda
- 深度學習原理
  - Neural network 基礎
  - Keras v.s. Tensorflow
- Keras處理手寫數字辨識資料集介
  - 建立Multiple Layer perceptron model (MLP)
  - 建立 convolutional neural network (CNN)
- Tensorflow手寫數字辨識資料集介
  - 建立Multiple Layer perceptron model (MLP)
  - 建立 convolutional neural network (CNN)

Neural network 基礎

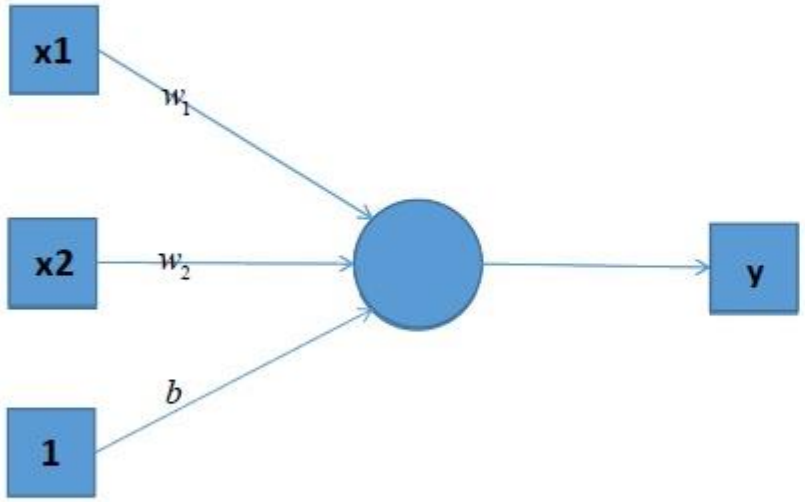




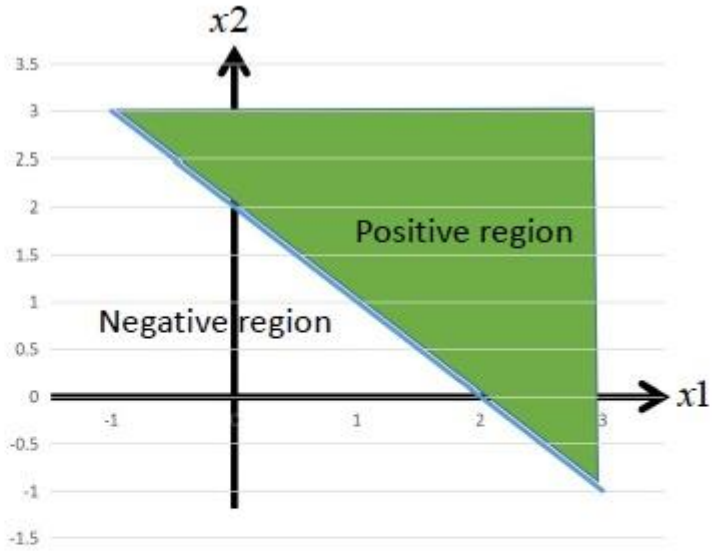
Perception



Perceptron



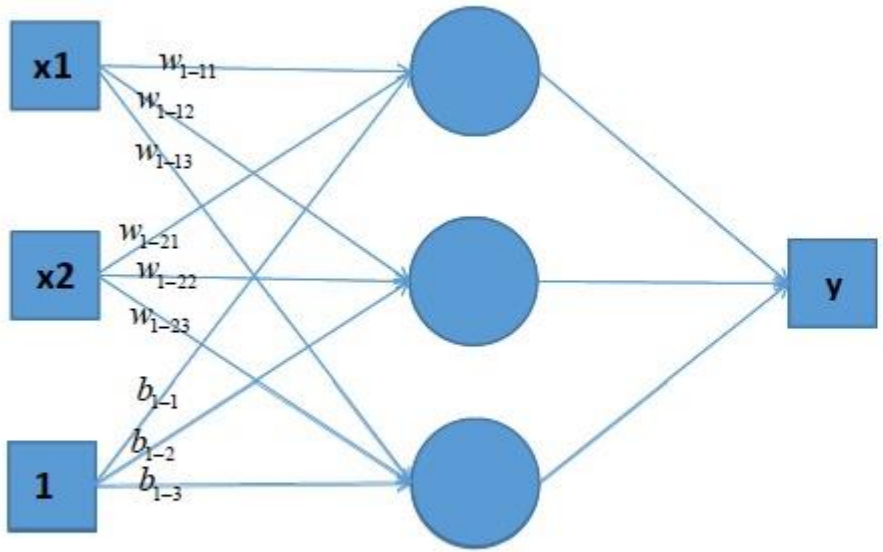
$$y = w_1 x_1 + w_2 x_2 + b$$



$$w_1 = 1, w_2 = 1, b = -2$$

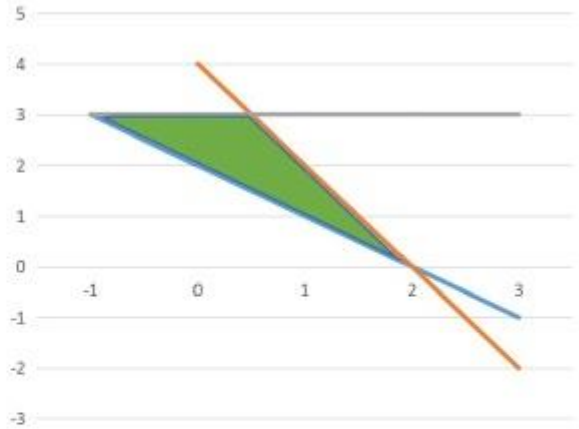
single layer perceptron is a linear classifier

Perceptron



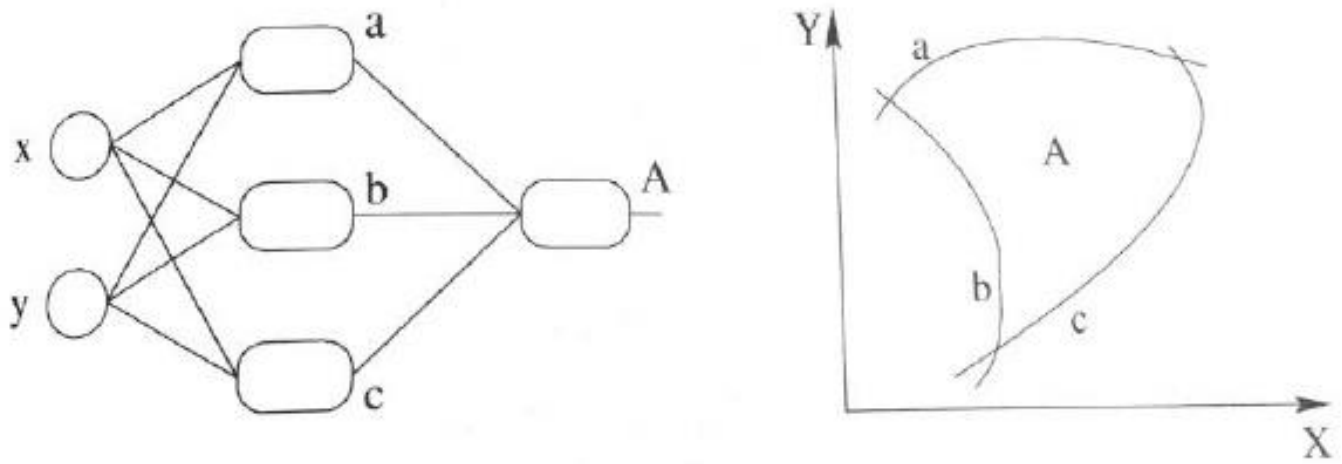
linear combination of three decision lines

single layer perceptron is a linear classifier



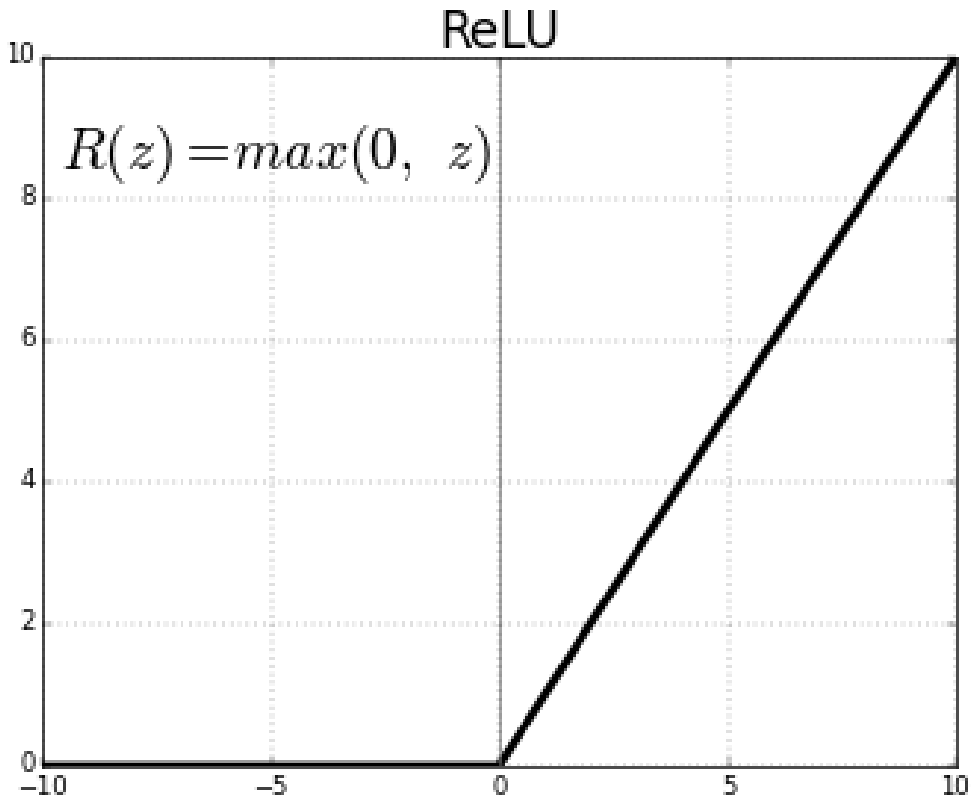
$$w_{1-11}=1, w_{1-12}=1, b_{1-1}=-2$$
$$w_{1-21}=2, w_{1-22}=1, b_{1-2}=4$$
$$w_{1-31}=0, w_{1-32}=1, b_{1-3}=3$$

Activation function

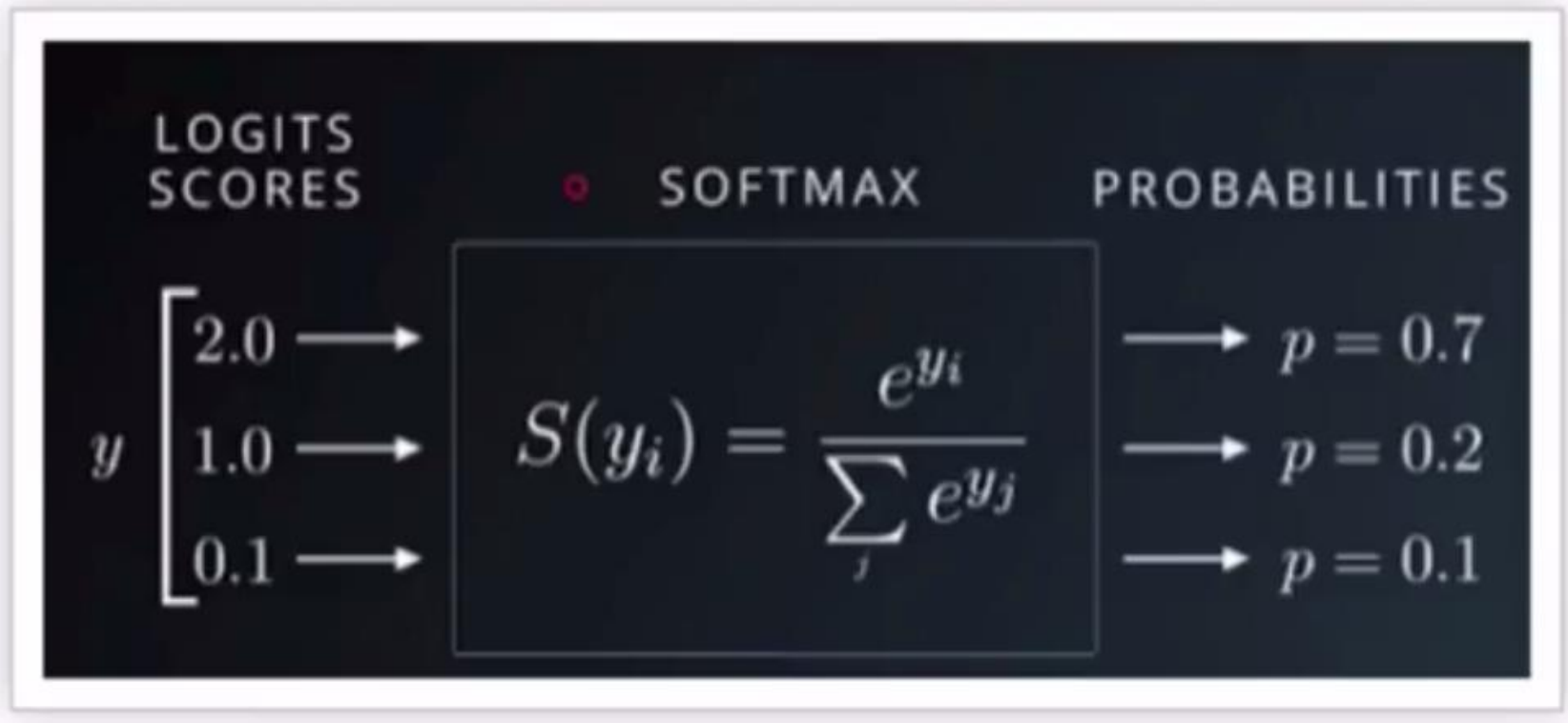


with sigmoid activation function

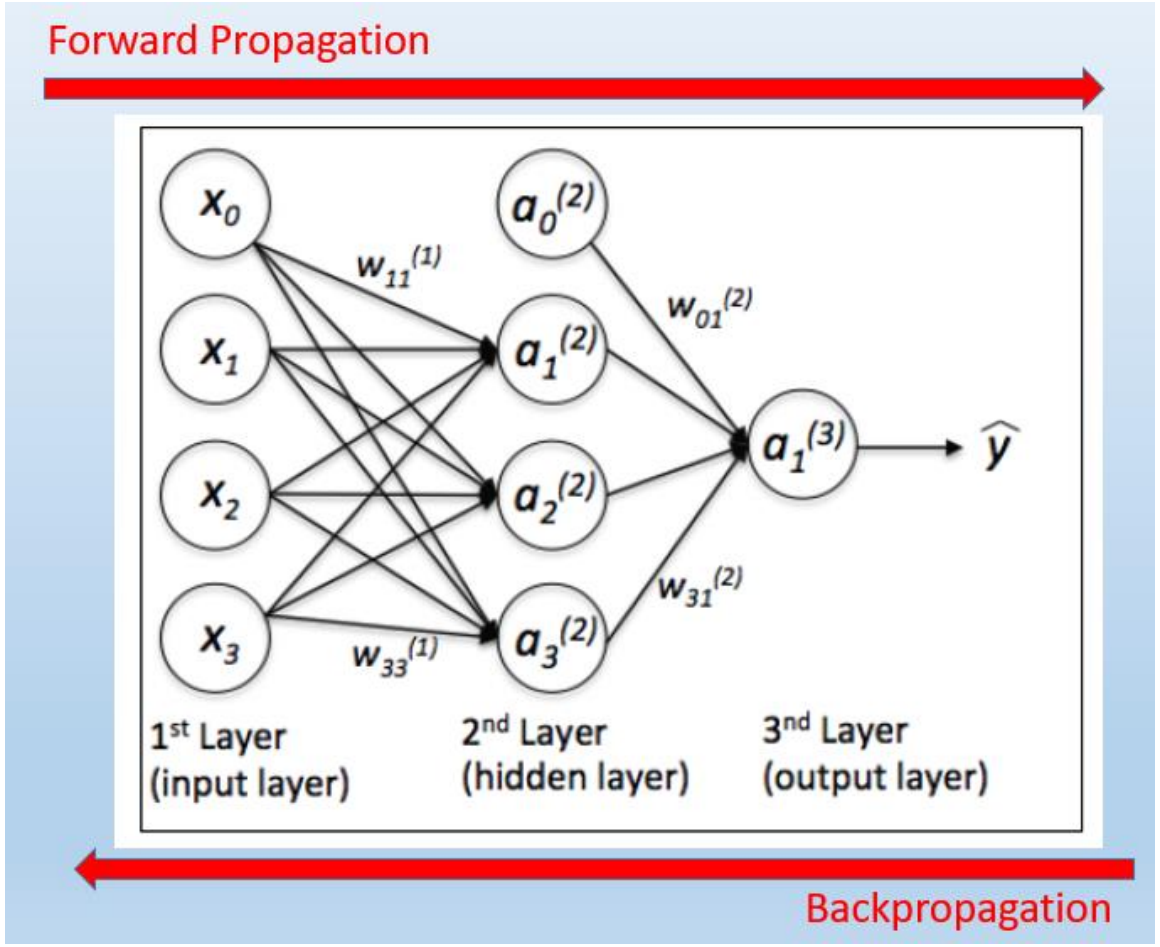
relu



softmax

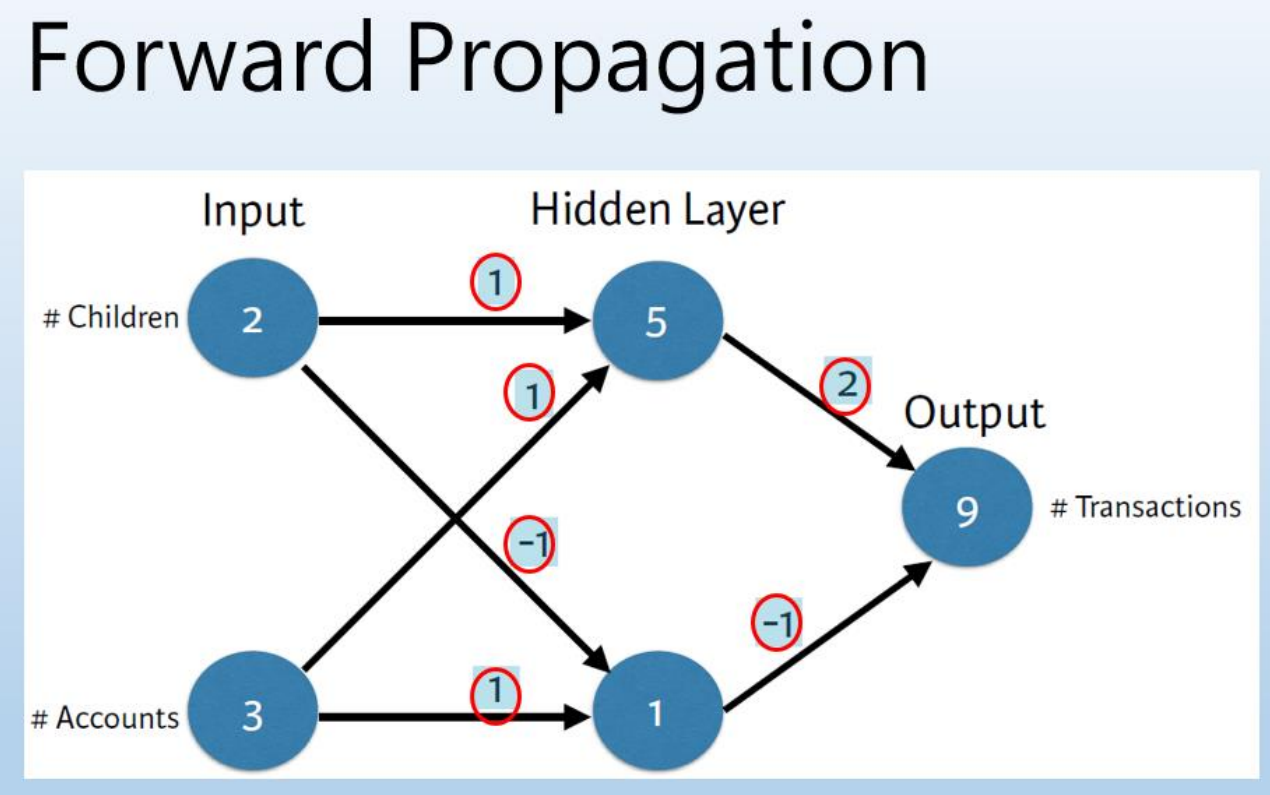

LOGITS
SCORES
SOFTMAX
PROBABILITIES
y
2.0
1.0
0.1
$S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}}$
$p = 0.7$
$p = 0.2$
$p = 0.1$

Forward & Back propagation



Forward Example

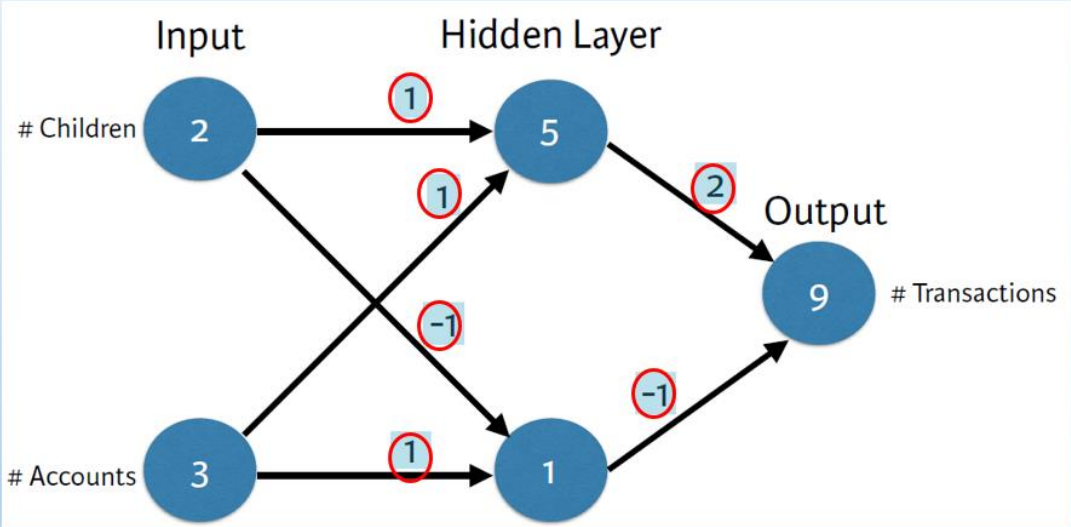
- Node 1 = 2 * 1 + 3 * 1 = 5,
- Node 2 = 2 * -1 + 3 * 1 = 1 ,
- Output = 5 * 2 + 1 * -1 = 9



Back propagation Example

- If Ouput = 13, Loss = 13-9 = 4
- Learning Rate = 0.01
- For Node 1
  - Gradient = -2 * 5 * 4 = -40
  - New Weight = 2 – 0.01*(-40) = 2.4
- For Node 2
  - Gradient = -2 * 1 * 4 = -8
  - New Weight = -1- 0.01*(-8) = -0.92

- 損失(Loss) = (y實際值 - y預測值) or (y預測值- y實際值)
- 梯度(Gradient) = -2 * input * (y實際值 - y 預測值)
- 調整後的權重 = 原權重 - (學習率 * 梯度)

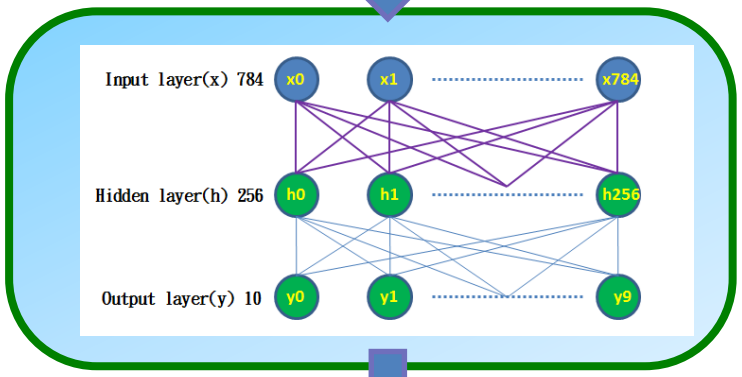


Dataset

Pre-Processing

Features Extraction

1. Input Features



2. Output from Model

Predictions

Labels (True Value)

3. Cross Entropy

Loss Function

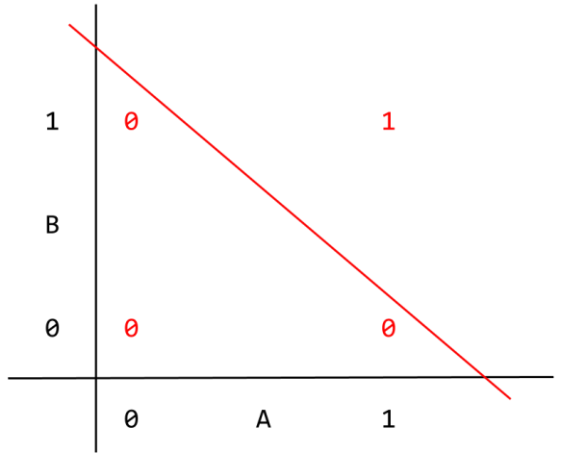
Error

Optimizer

4. Update Weights and Bias

Neural Network for XOR Gate

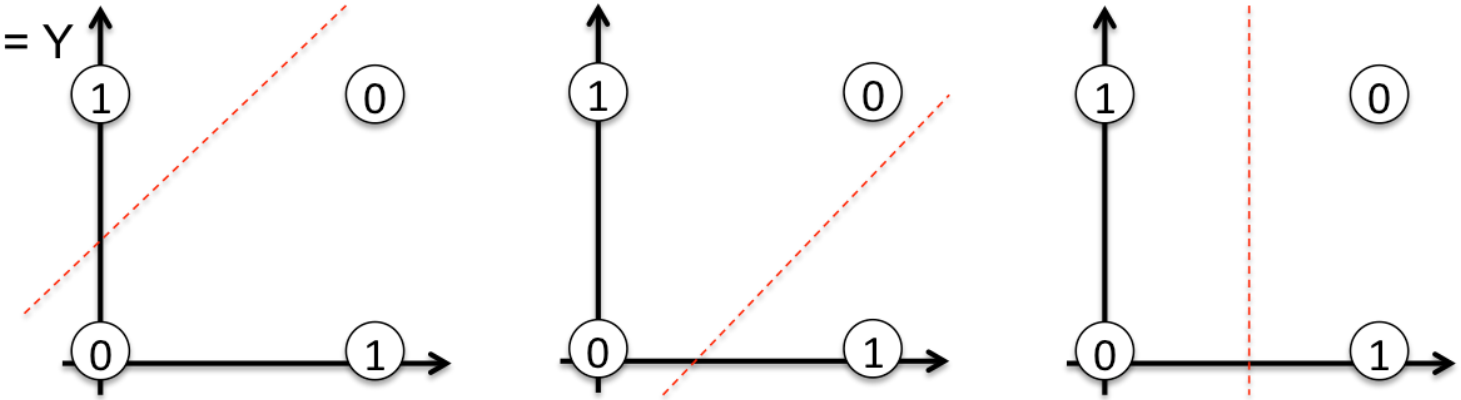
- AND is **linearly separable**



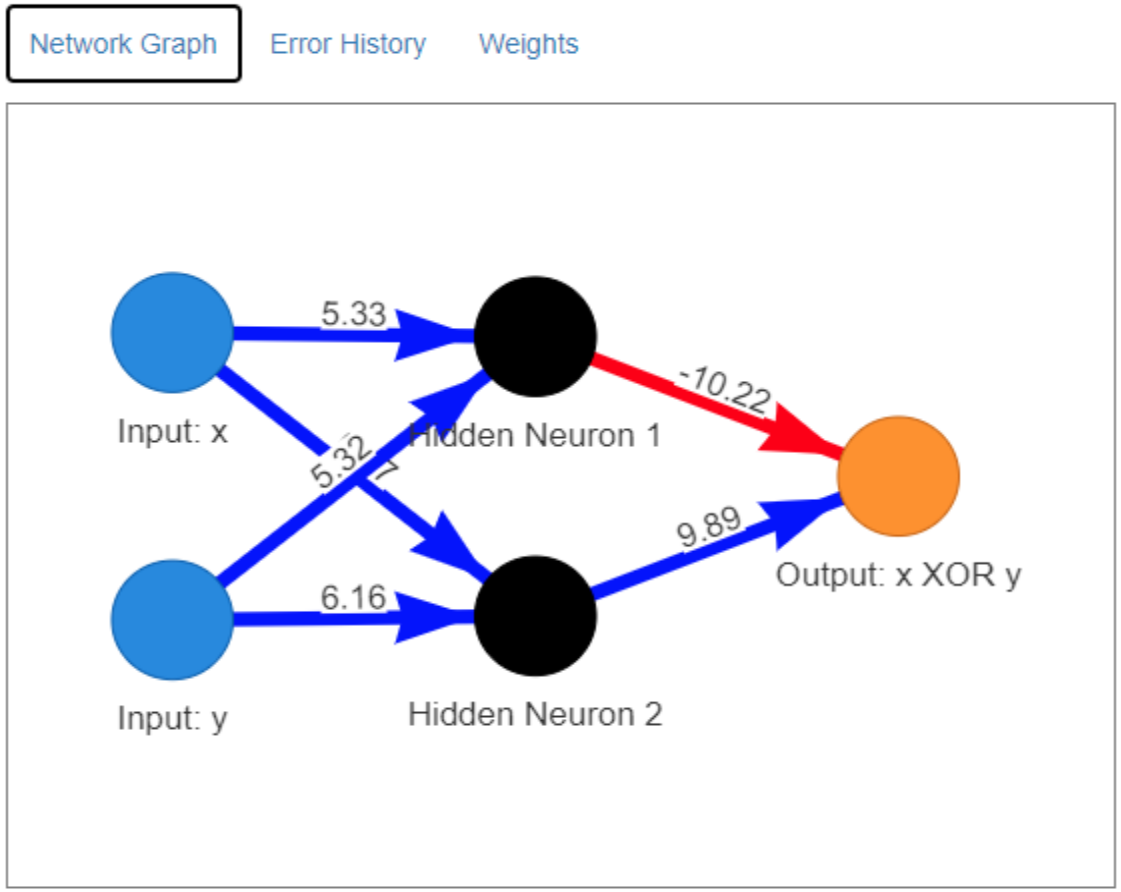
- XOR is not **non-linearly separable**

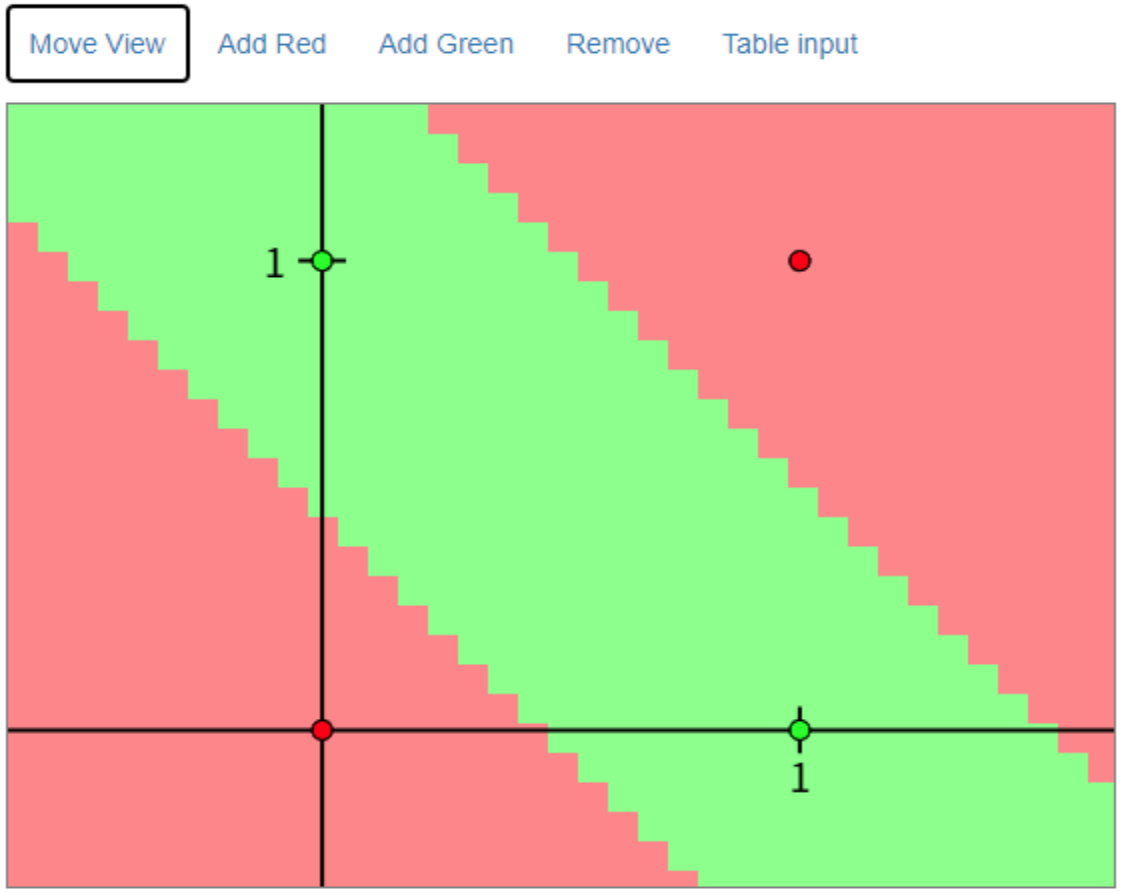
XOR(x_1>0, x_2>0) = Y

x_1	x_2	Y
0	0	0
0	1	1
1	0	1
1	1	0



- https://lecture-demo.ira.uka.de/neural-network-demo/?preset=Binary%20Classifier%20for%20XOR





古早時代的程式

```
def fit(self, data, labels, learning_rate=0.1, epochs=100):

    # Add bias units to the input layer -
    # add a "1" to the input data (the always-on bias neuron)
    ones = numpy.ones((1, data.shape[0]))
    Z = numpy.concatenate((ones.T, data), axis=1)

    for k in range(epochs):
        if (k+1) % 10000 == 0:
            print('epochs: {}'.format(k+1))

        sample = numpy.random.randint(X.shape[0])

        # We will now go ahead and set up our feed-forward propagation:
        x = [Z[sample]]
        y = self._forward_prop(x)

        # Now we do our back-propagation of the error to adjust the weights:
        target = labels[sample]
        self._back_prop(y, target, learning_rate)
```

TensorFlow vs. Keras

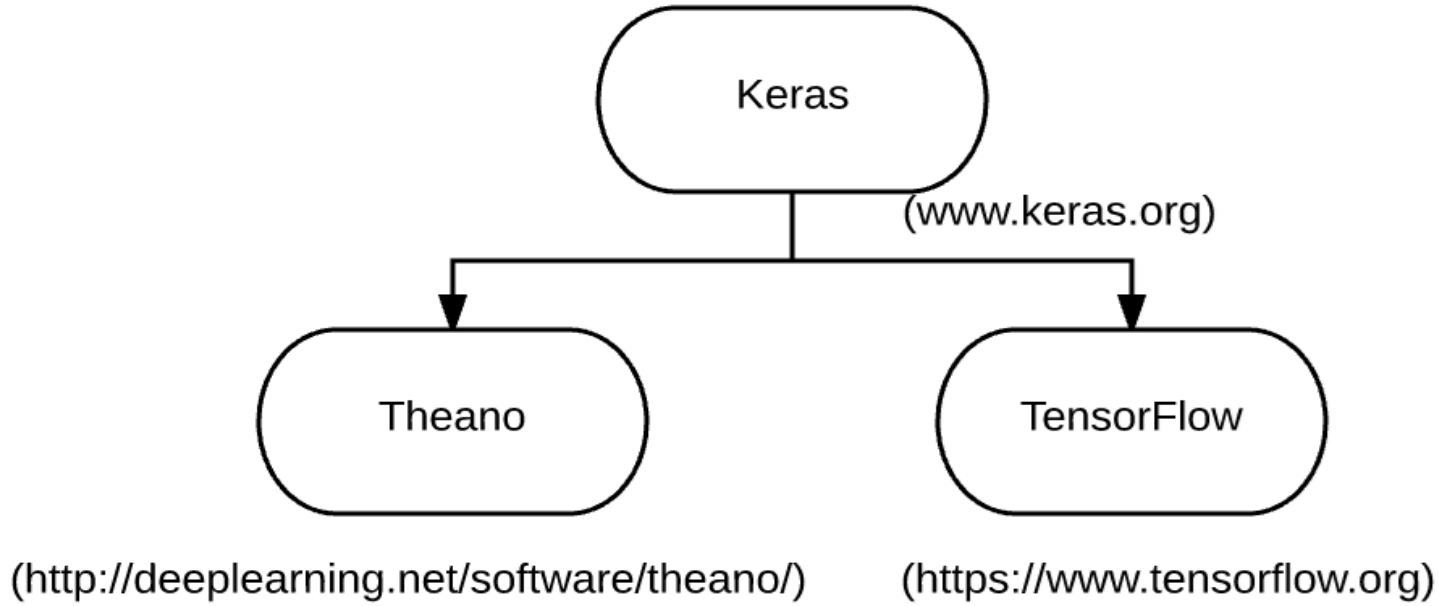
	Keras	TensorFlow
學習難易度	簡單	比較困難
使用彈性	中等	高
開發生產力	高	中等
執行效能	高	高
適合使用者	初學者	進階使用者
張量(矩陣)運算	不需自行設計	需自行設計

課程大綱

- Python 基礎
  - Python、Numpy、Matplot、Panda
- 深度學習原理
  - Neural network 基礎
  - Keras v.s. Tensorflow
- Keras處理手寫數字辨識資料集介
  - 建立Multiple Layer perceptron model (MLP)
  - 建立 convolutional neural network (CNN)
- Tensorflow手寫數字辨識資料集介
  - 建立Multiple Layer perceptron model (MLP)
  - 建立 convolutional neural network (CNN)

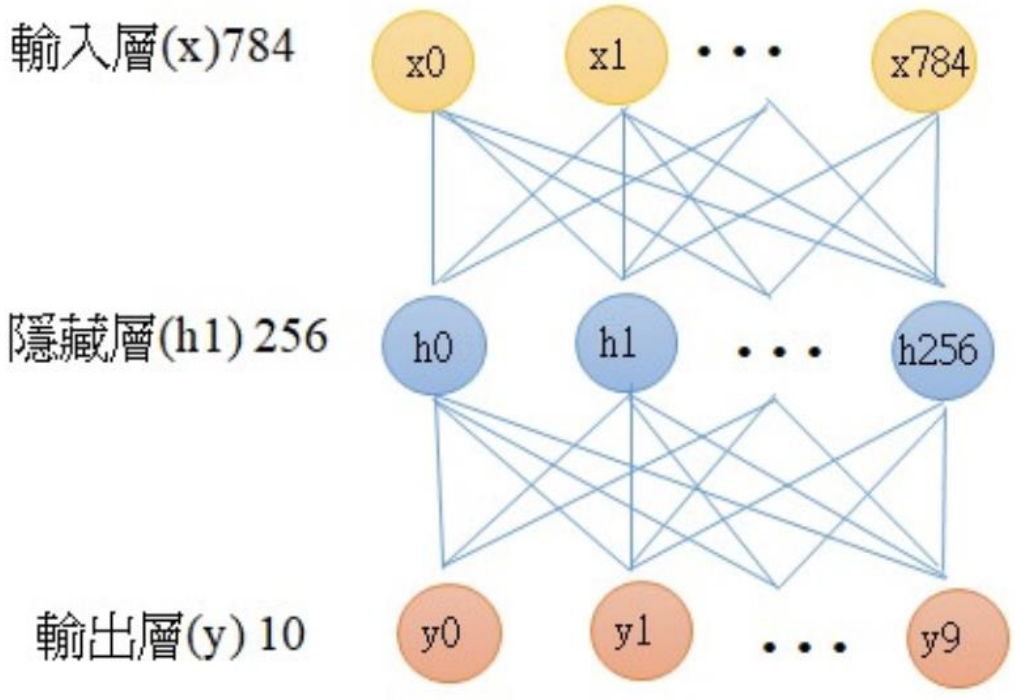
Keras – 簡介

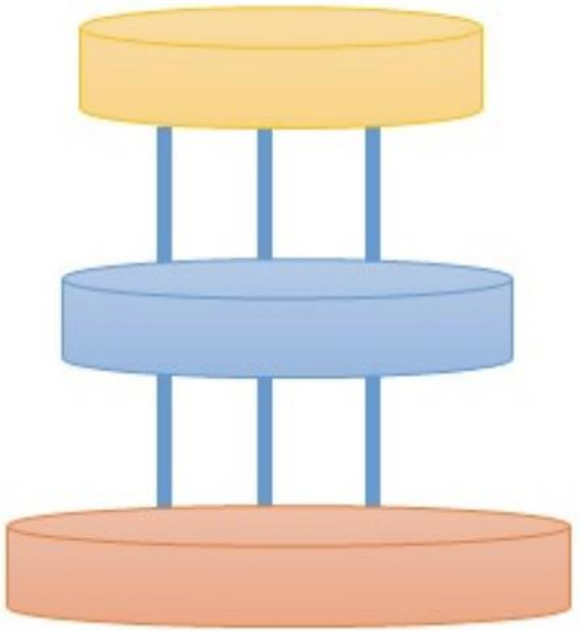
- Keras是一個開放原始碼，高階深度學習程式庫，使用Python編寫，能夠運行在TensorFlow或Theano之上。
- 建立模型、訓練、預測



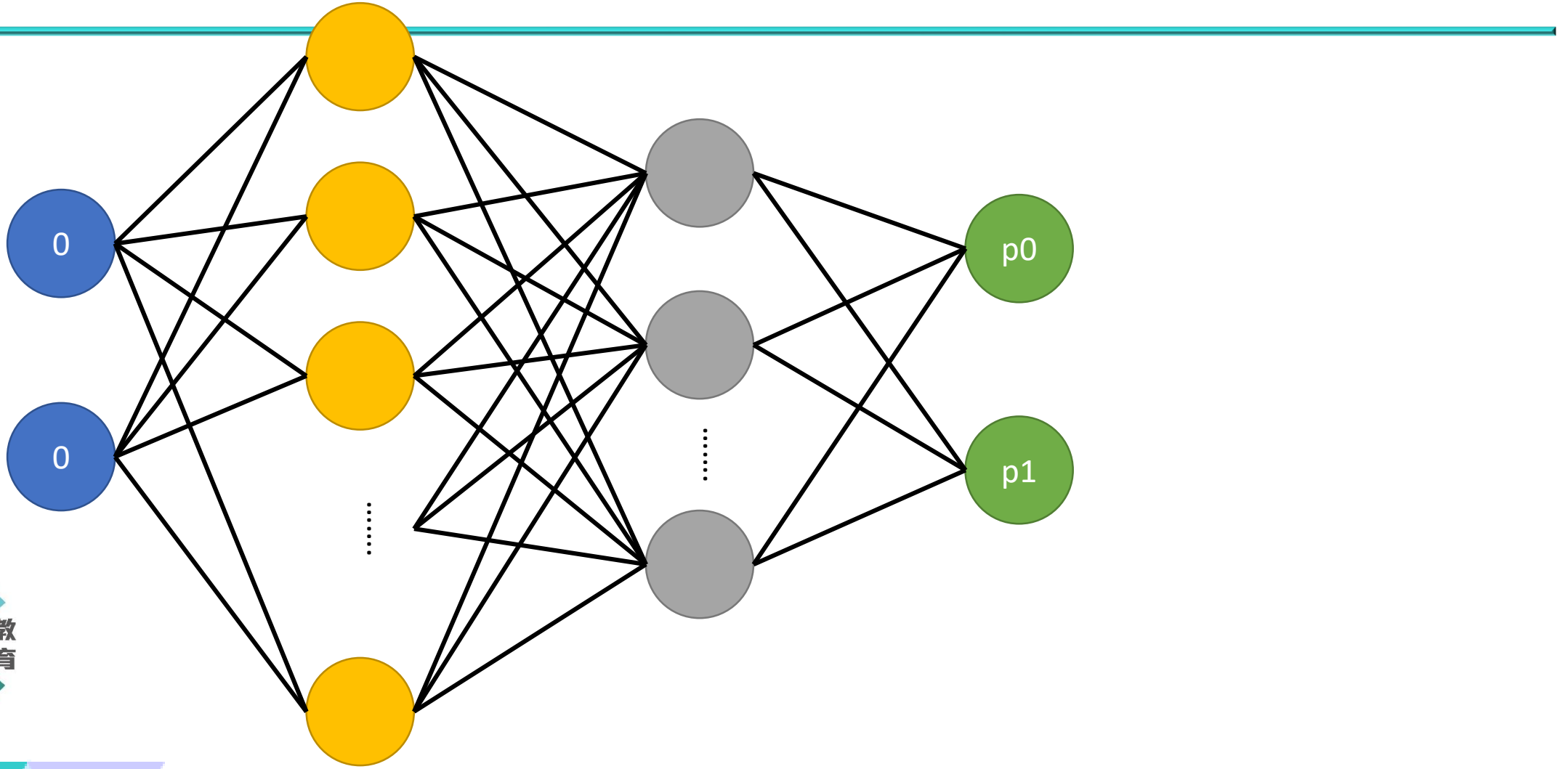
Keras – Model Level

- Model level深度學習程式庫
  - 只需要處理建立模型、訓練、預測





Neural Network for XOR Gate



使用keras

- 建立

```
model = Sequential()
```

```
model.add(Dense(10,input_dim=2, activation='relu'))
```

```
model.add(Dense(6, activation='relu'))
```

```
model.add(Dense(2, activation='softmax'))
```

```
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

- 訓練

```
model.fit(x,y,epochs=5000,batch_size=100)
```

```
Epoch 1/5000
4/4 [==============================] - 0s 58ms/step - loss: 1.3012 - acc: 0.5000
Epoch 2/5000
4/4 [==============================] - 0s 251us/step - loss: 1.2958 - acc: 0.5000
Epoch 3/5000
4/4 [==============================] - 0s 202us/step - loss: 1.2887 - acc: 0.5000
Epoch 4/5000
4/4 [==============================] - 0s 186us/step - loss: 1.2815 - acc: 0.5000
Epoch 5/5000
```

- 預測

```
model.predict(np.array([[0,1]]))
```

```
array([[0.00924953, 0.9907505 ]], dtype=float32)
```

AI 教育

Keras - Labs

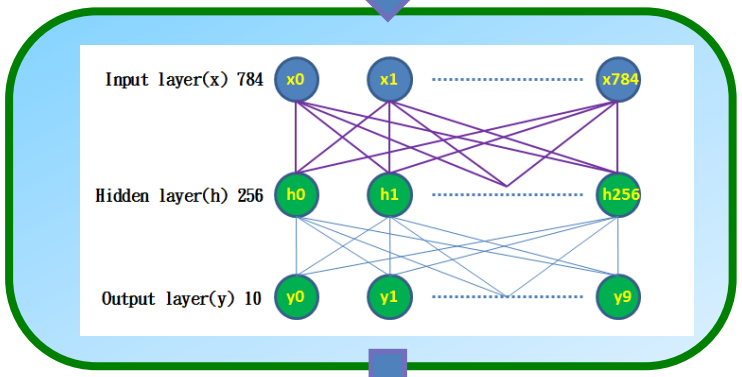
- Keras多元感知器(MLP)辨識手寫數字
  - MNIST-MLP.ipynb

Dataset

Pre-Processing

Features Extraction

1. Input Features



2. Output from Model

Predictions

Labels (True Value)

3. Cross Entropy

Loss Function

Error

Optimizer

4. Update Weights and Bias

MNIST dataset

MNIST-MLP.ipynb

- **資料預處理：**

```
from keras.datasets import mnist
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()
```

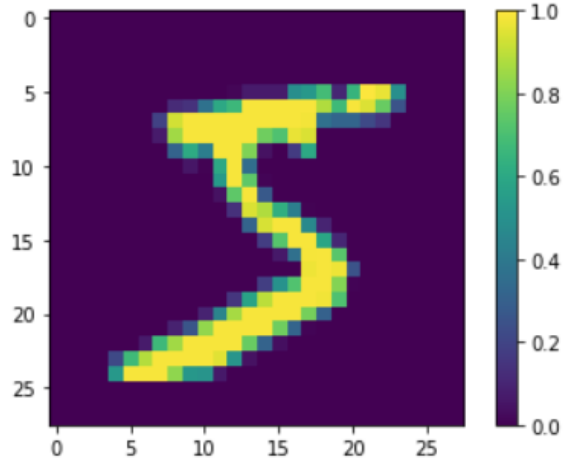
```
train_images.shape
```

```
(60000, 28, 28)
```

```
train_labels.shape
```

```
(60000,)
```

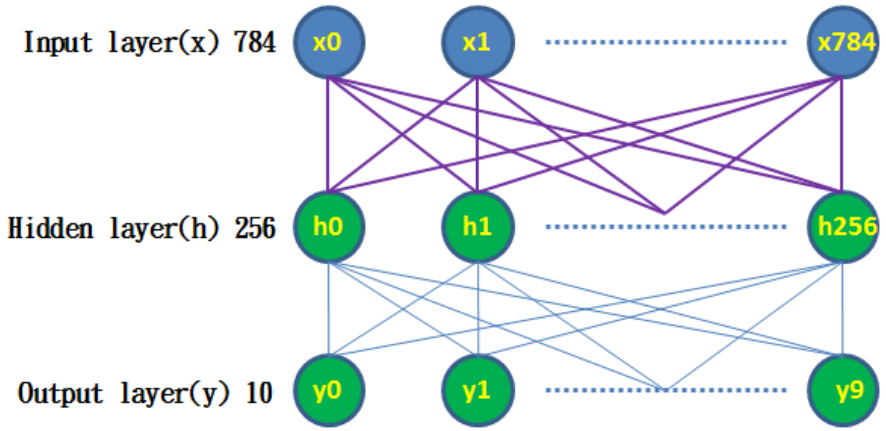
```
train_images = train_images / 255.0
test_images = test_images / 255.0
plot_single(train_images[0])
```



- **建立模型：**
  - Input Layer　：有 28x28=784 個神經元
  - Hidden layers：共有 256 個神經元
  - Output layer　：共有 10 個 神經元

```
model = Sequential()
model.add(Flatten(input_shape=(28, 28)))
model.add(Dense(256, activation=tf.nn.relu, kernel_initializer='normal'))
model.add(Dense(10, activation=tf.nn.softmax, kernel_initializer='normal'))
```

```
model.compile(optimizer=tf.train.AdamOptimizer(),
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```



- Flatten(): Flattens the input
- Dense(): 建立Fully Connected network

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten_2 (Flatten)          (None, 784)               0
_________________________________________________________________
dense_3 (Dense)              (None, 256)               200960
_________________________________________________________________
dense_4 (Dense)              (None, 10)                2570
=================================================================
Total params: 203,530
Trainable params: 203,530
Non-trainable params: 0
_________________________________________________________________
None
```

- 訓練模型：

```
train_history = model.fit(train_images, train_labels, validation_split=0.2, epochs=10, batch_size=200)
```

```
Train on 48000 samples, validate on 12000 samples
Epoch 1/10
48000/48000 [==============================] - 1s 17us/step - loss: 0.0813 - acc: 0.9749 - val_loss: 0.0788 - val_acc: 0.9777
Epoch 2/10
48000/48000 [==============================] - 1s 17us/step - loss: 0.0773 - acc: 0.9764 - val_loss: 0.0774 - val_acc: 0.9773
Epoch 3/10
48000/48000 [==============================] - 1s 17us/step - loss: 0.0746 - acc: 0.9764 - val_loss: 0.0751 - val_acc: 0.9788
```

參數說明

* x=train_images: features 數字的影像特徵值 (60,000 x 784 的陣列).
* y=train_labels: label 數字的 One-hot encoding 陣列 (60,000 x 10 的陣列)
* validation_split = 0.2: 設定訓練資料與 cross validation 的資料比率. 也就是說會有 0.8 * 60,000 = 48,000 作為訓練資料; 0.2 * 60,000 = 12,000 作為驗證資料.
* epochs = 10: 執行 10 次的訓練週期.
* batch_size = 200: 每一批次的訓練筆數為 200

- 評估模型準確率：

```
scores = model.evaluate(test_images, test_labels)
print("Accuracy of testing data = {:2.1f}%".format(scores[1]*100.0))
```

```
10000/10000 [==============================] - 0s 14us/step
Accuracy of testing data = 98.1%
```

- 預測：

```
predictions = model.predict(test_images)
print predictions.shape
print("\n")
print predictions[0]
```

```
(10000, 10)


[5.5651244e-09 1.5294382e-10 4.6140903e-07 5.0855342e-05 4.9134763e-15
 4.5012856e-09 3.9966551e-15 9.9994826e-01 9.3465147e-09 3.4660687e-07]
```

```
np.argmax(predictions[0])
```

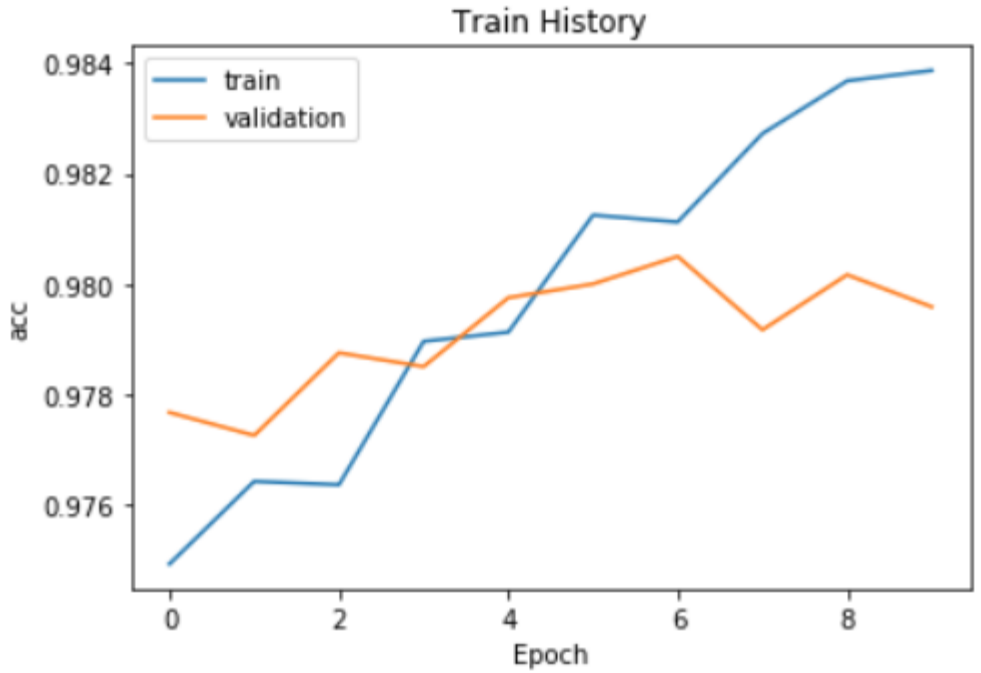
```
7
```

```
test_labels[0]
```

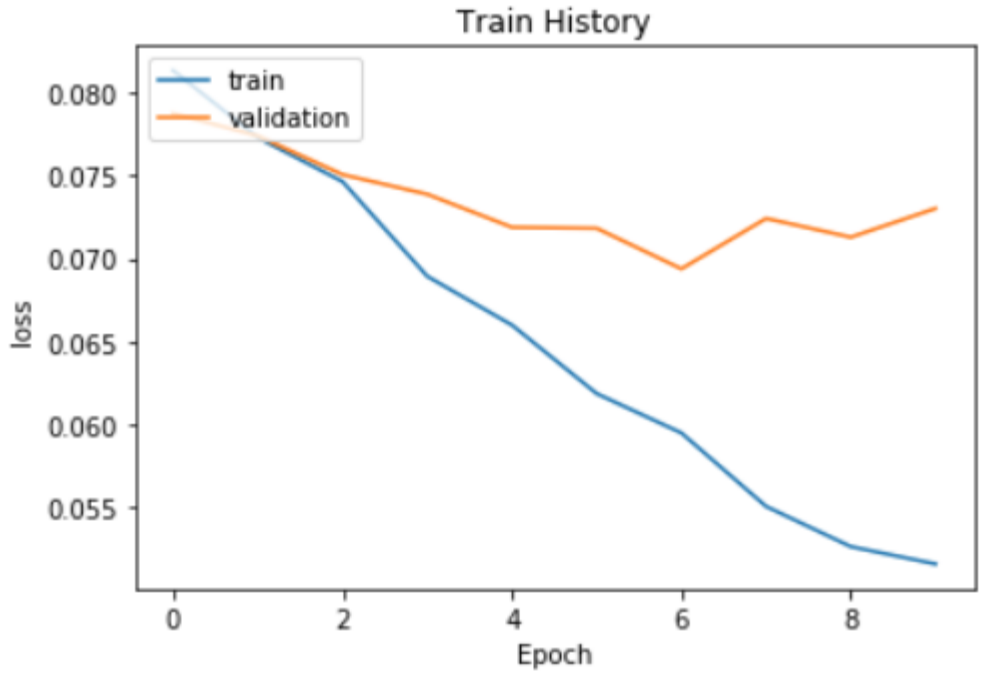
```
7
```

- Overfitting:

```
show_train_history(train_history,'acc','val_acc')
```




```
show_train_history(train_history,'loss','val_loss')
```



- 修正模型

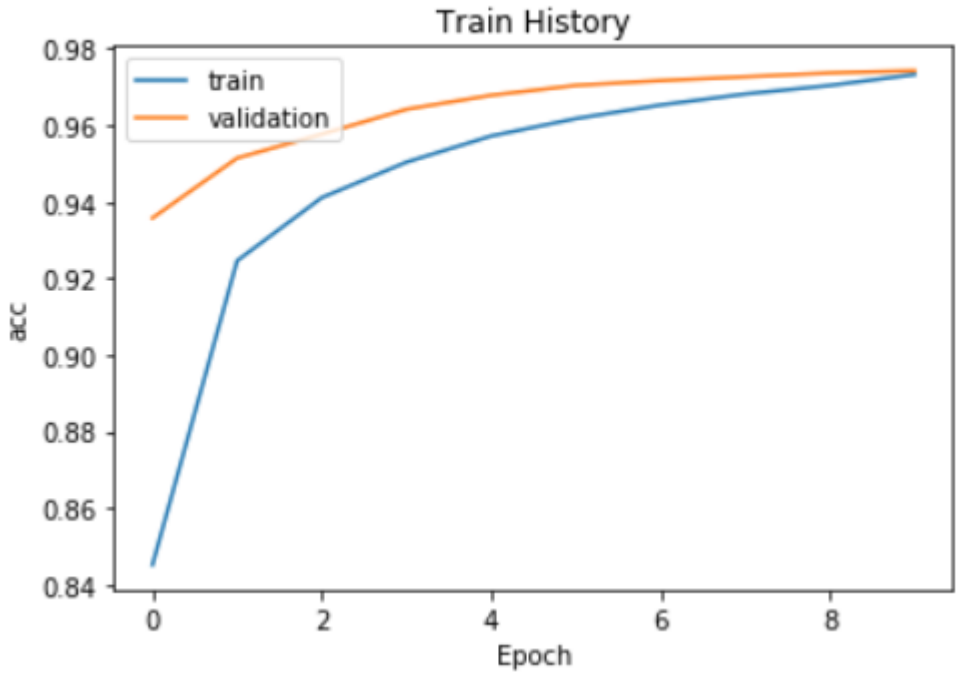
```
model = Sequential()
model.add(Flatten(input_shape=(28, 28)))
model.add(Dense(256, activation=tf.nn.relu, kernel_initializer='normal'))
model.add(Dropout(0.5))
model.add(Dense(10, activation=tf.nn.softmax, kernel_initializer='normal'))
```

```
model.compile(optimizer=tf.train.AdamOptimizer(),
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

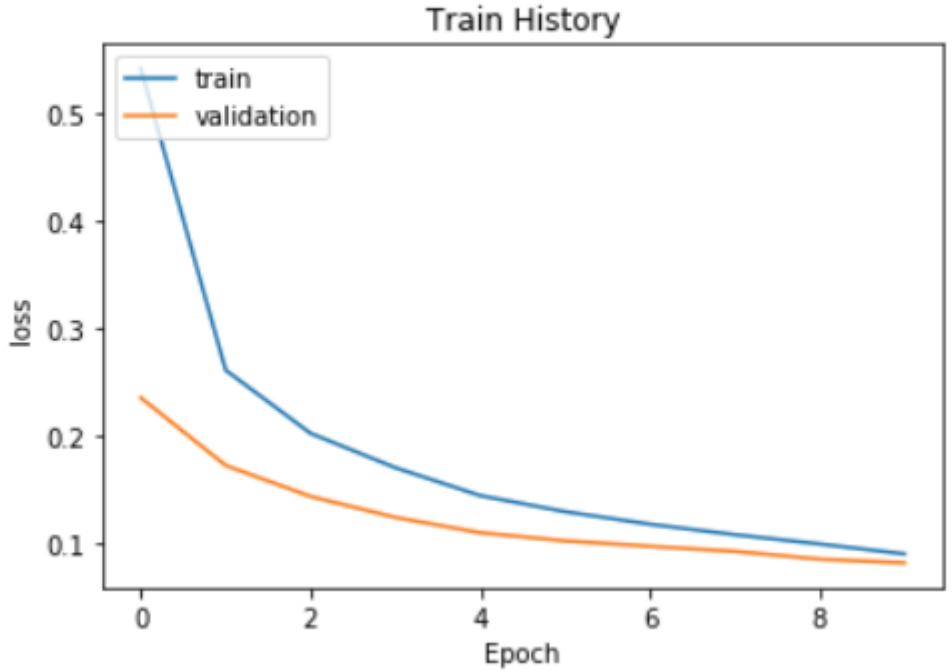
```
print(model.summary())
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten_2 (Flatten)          (None, 784)               0
_________________________________________________________________
dense_3 (Dense)              (None, 256)               200960
_________________________________________________________________
dropout_2 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_4 (Dense)              (None, 10)                2570
=================================================================
Total params: 203,530
Trainable params: 203,530
Non-trainable params: 0
_________________________________________________________________
None
```

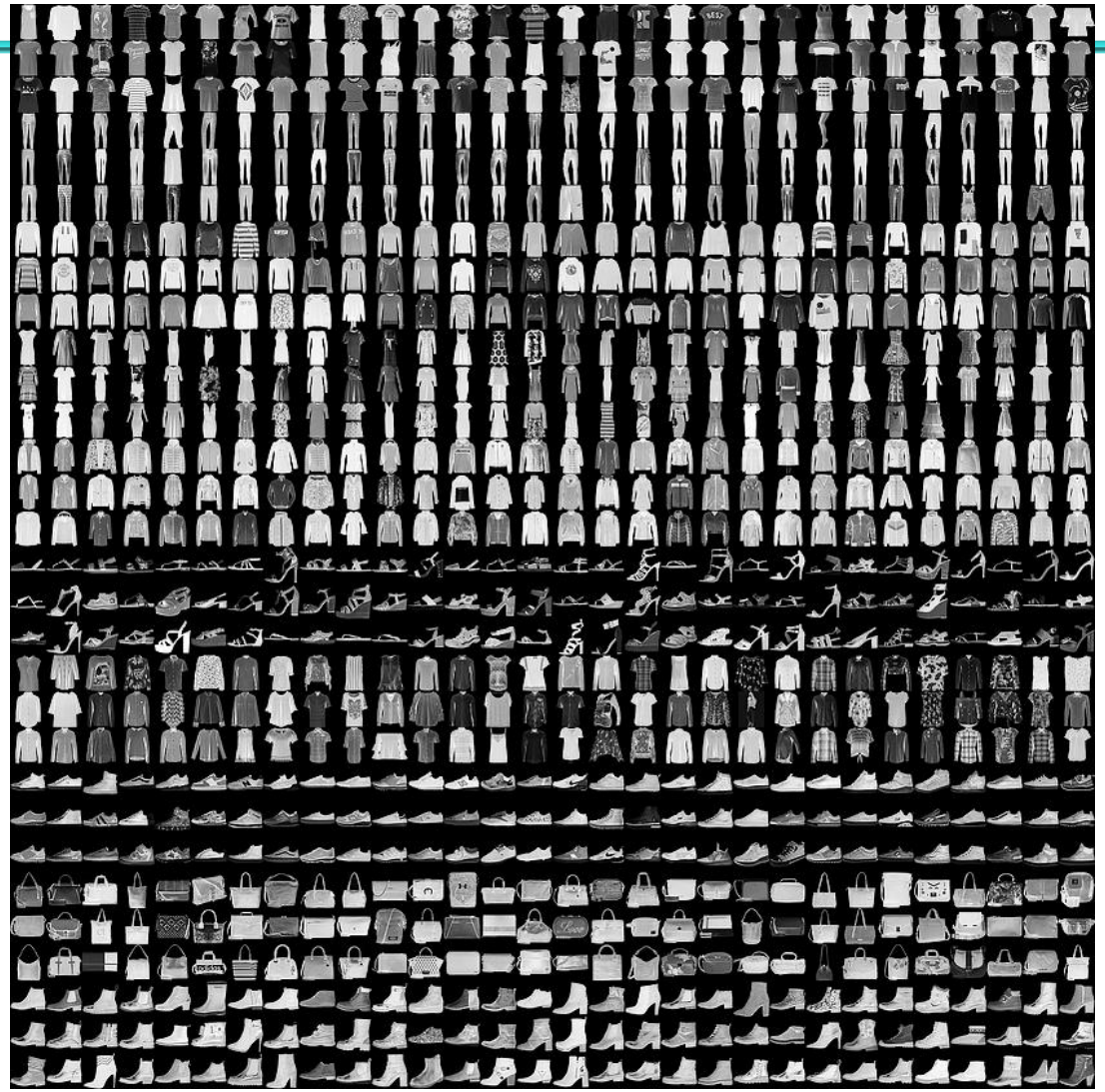
```
show_train_history(train_history,'acc','val_acc')
```




```
show_train_history(train_history,'loss','val_loss')
```



Exercise: fashion MNIST



課程大綱

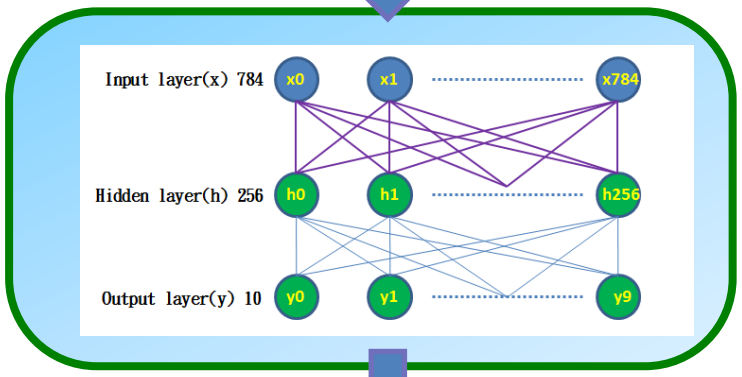
- Python 基礎
  - Python、Numpy、Matplot、Panda
- 深度學習原理
  - Neural network 基礎
  - Keras v.s. Tensorflow
- Keras處理手寫數字辨識資料集介
  - 建立Multiple Layer perceptron model (MLP)
  - 建立 convolutional neural network (CNN)
- Tensorflow手寫數字辨識資料集介
  - 建立Multiple Layer perceptron model (MLP)
  - 建立 convolutional neural network (CNN)

Dataset

Pre-Processing

Features Extraction
(Convolution)

1. Input Features



2. Output from Model

Predictions

Labels (True Value)

3. Cross Entropy

Loss Function

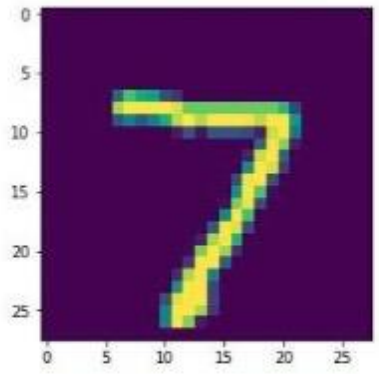
Error

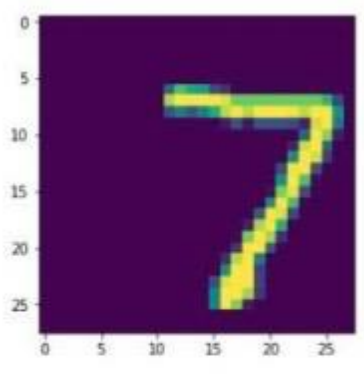
Optimizer

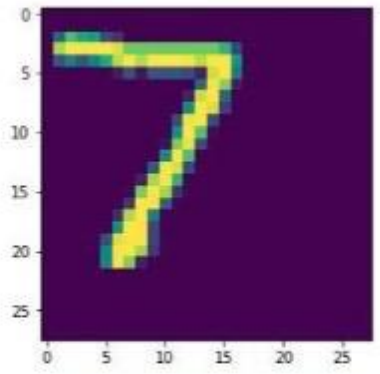
4. Update Weights and Bias

Keras - CNN

- MLP是計算784個pixel間的關係去預測數字，並沒有做任何的Features Extraction
- 若圖片進行平移後，準確性會下降







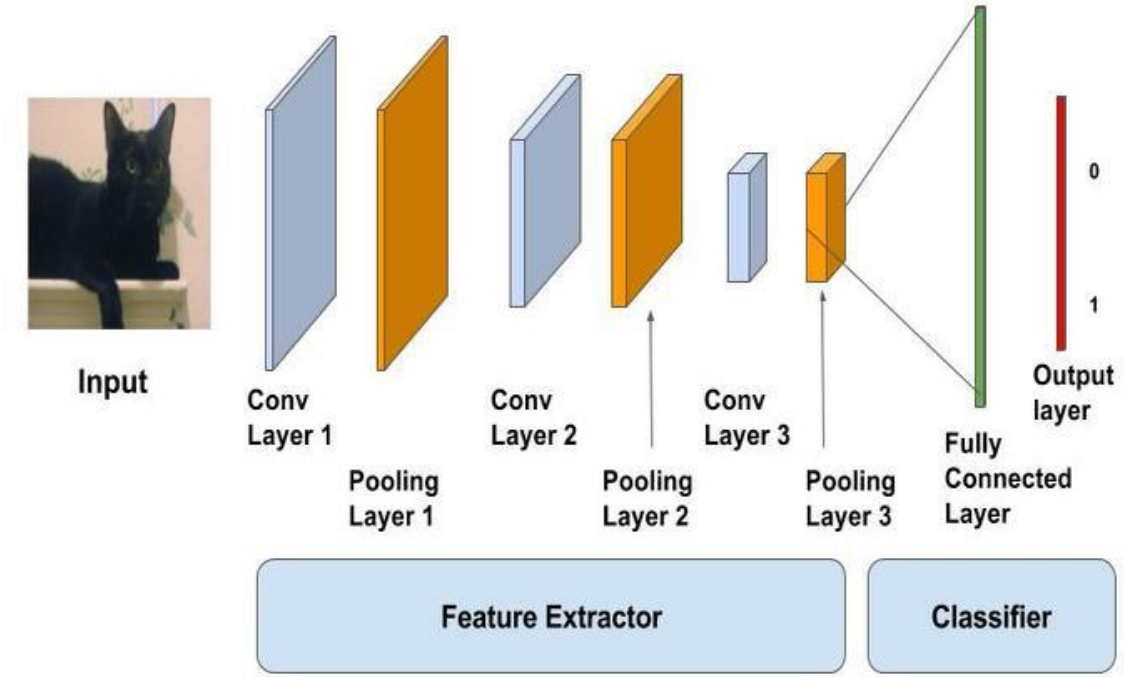
```
from util.my_plot import generate_shift_mnist_data

shift_img, shift_img_label = generate_shift_mnist_data(10000)
plot_batch(shift_img, shift_img_label,class_names, 5)
```


```
scores = model.evaluate(shift_img, shift_img_label)
print("Accuracy of generated shift data = {:2.1f}%".format(scores[1]*100.0))
```

```
10000/10000 [==============================] - 0s 14us/step
Accuracy of generated shift data = 50.8%
```

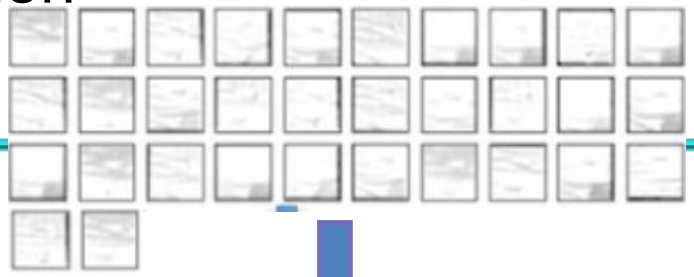
Keras - CNN

- 取特徵後，再丟給MLP
  - Convolution Layer
  - Max Pooling Layer



- 28*28
- 1 image

The first convolution

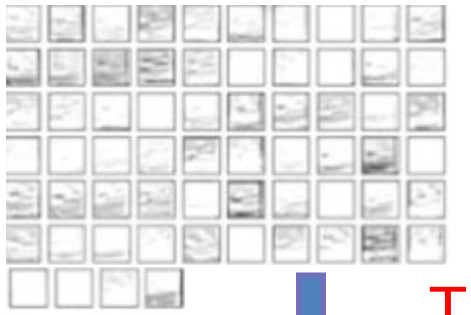


- 28*28
- 16 images (or feature map)

The first pooling

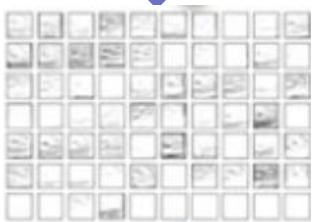
- 14*14
- 16 images

The second convolution



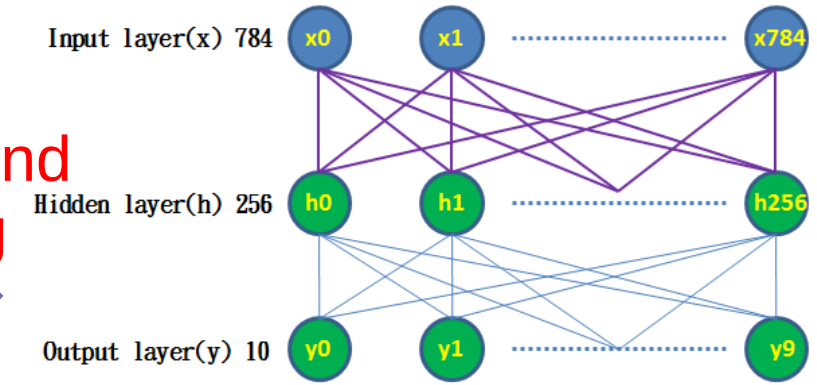
- 14*14
- 36 images

The second pooling



- 7*7
- 36 images

FC



CNN – Convolution

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

stride=1

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

3	-1	-3	-1
-3	1	0	-3
-3	-3	0	1
3	-2	-2	-1

CNN – Convolution

Filter 1

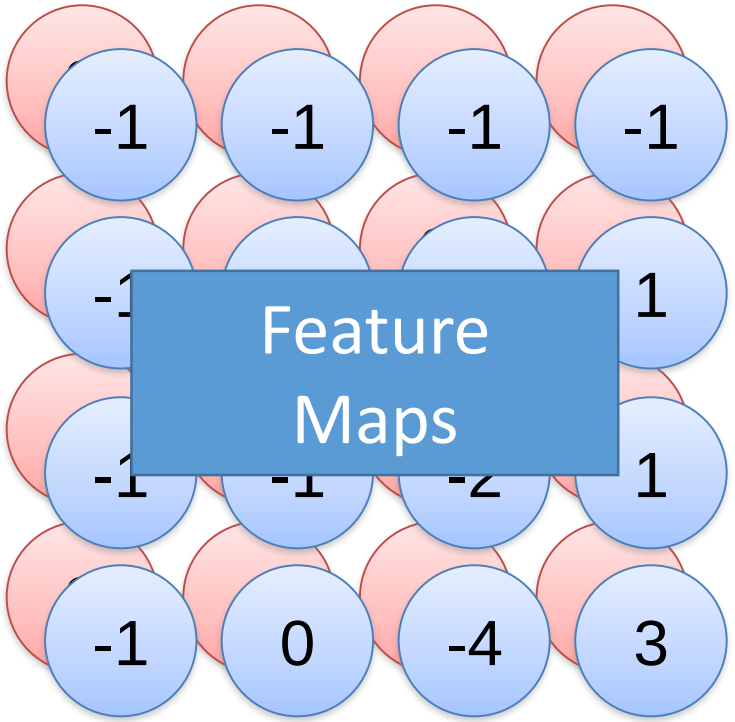
-1	1	-1
-1	1	-1
-1	1	-1

Filter 2

stride=1

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image



4 x 4 image

選擇檔案 未選擇任何檔案 Live video

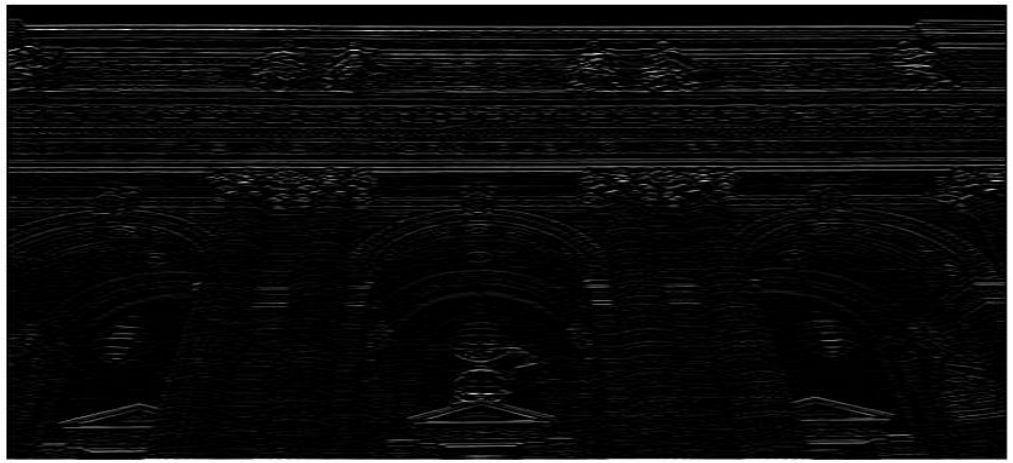
-1	2	-1
-1	2	-1
-1	2	-1

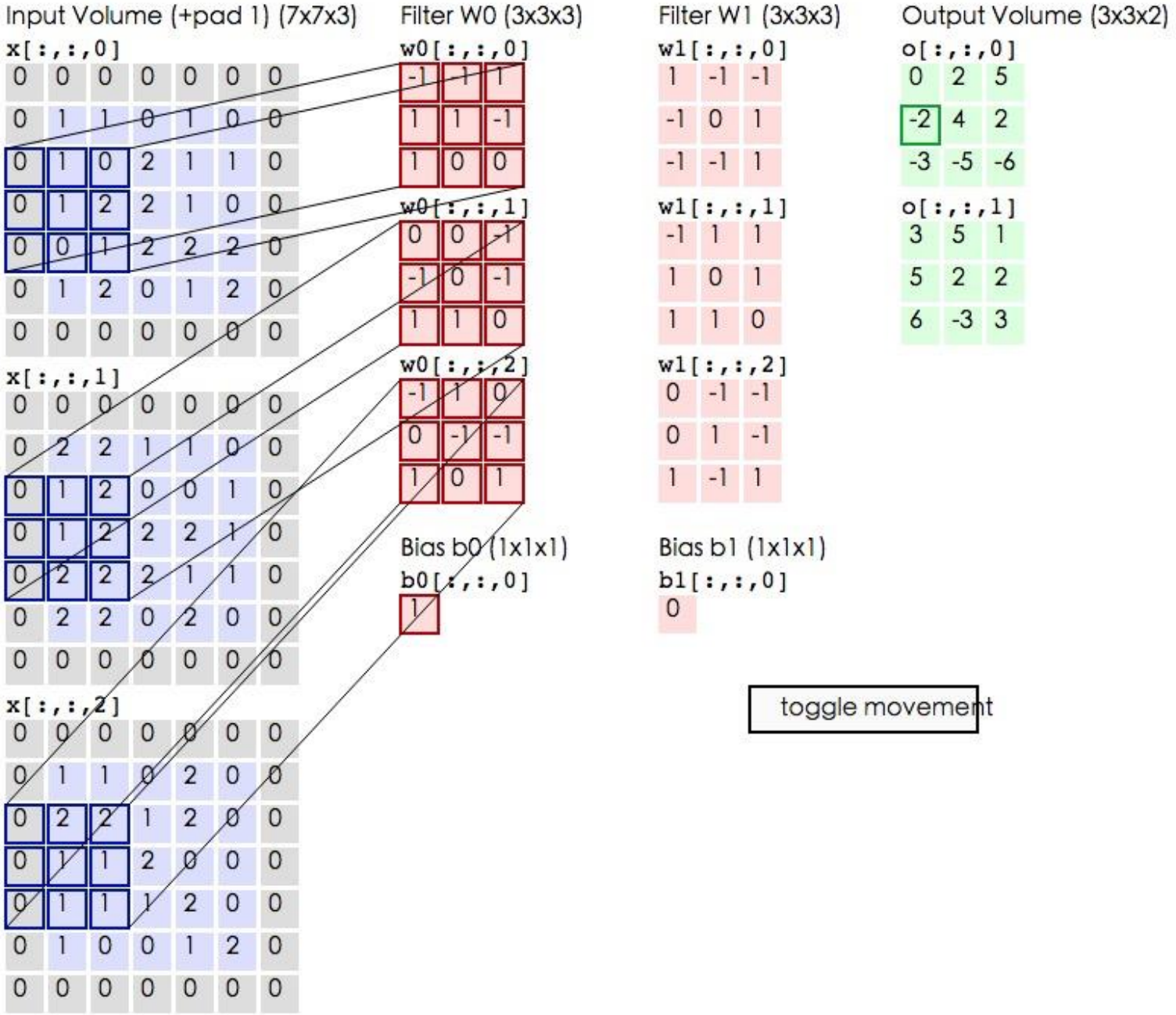
custom

選擇檔案 未選擇任何檔案 Live video

-1	-1	-1
2	2	2
-1	-1	-1

custom




Input Volume (+pad 1) (7x7x3)
x[:,:,0]
0 0 0 0 0 0 0
0 1 1 0 1 0 0
0 1 0 2 1 1 0
0 1 2 2 1 0 0
0 0 1 2 2 2 0
0 1 2 0 1 2 0
0 0 0 0 0 0 0
x[:,:,1]
0 0 0 0 0 0 0
0 2 2 1 1 0 0
0 1 2 0 0 1 0
0 1 2 2 2 1 0
0 2 2 2 1 1 0
0 2 2 0 2 0 0
0 0 0 0 0 0 0
x[:,:,2]
0 0 0 0 0 0 0
0 1 1 0 2 0 0
0 2 2 1 2 0 0
0 1 1 2 0 0 0
0 1 1 1 2 0 0
0 1 0 0 1 2 0
0 0 0 0 0 0 0
Filter W0 (3x3x3)
w0[:,:,0]
-1 -1 1
1 1 -1
1 0 0
w0[:,:,1]
0 0 -1
-1 0 -1
1 1 0
w0[:,:,2]
-1 1 0
0 -1 -1
1 0 1
Bias b0 (1x1x1)
b0[:,:,0]
1
Filter W1 (3x3x3)
w1[:,:,0]
1 -1 -1
-1 0 1
-1 -1 1
w1[:,:,1]
-1 1 1
1 0 1
1 1 0
w1[:,:,2]
0 -1 -1
0 1 -1
1 -1 1
Bias b1 (1x1x1)
b1[:,:,0]
0
Output Volume (3x3x2)
o[:,:,0]
0 2 5
-2 4 2
-3 -5 -6
o[:,:,1]
3 5 1
5 2 2
6 -3 3
toggle movement

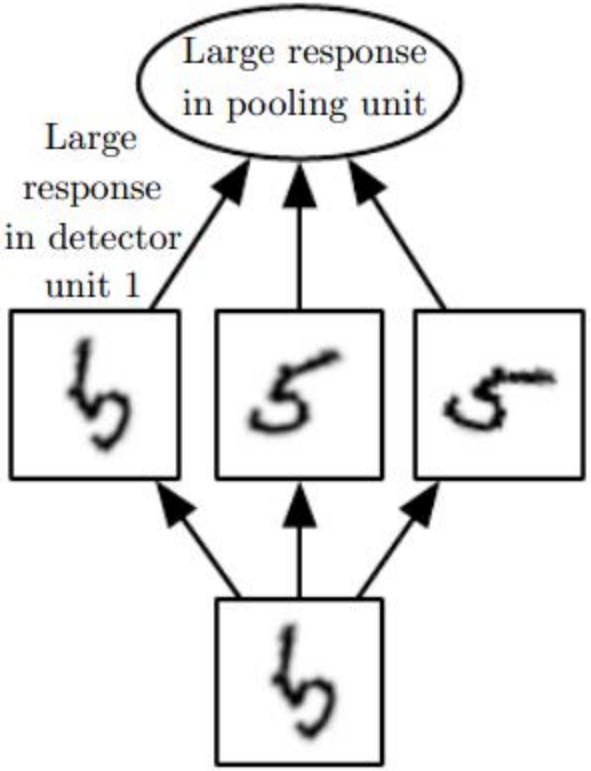
CNN – Max Pooling

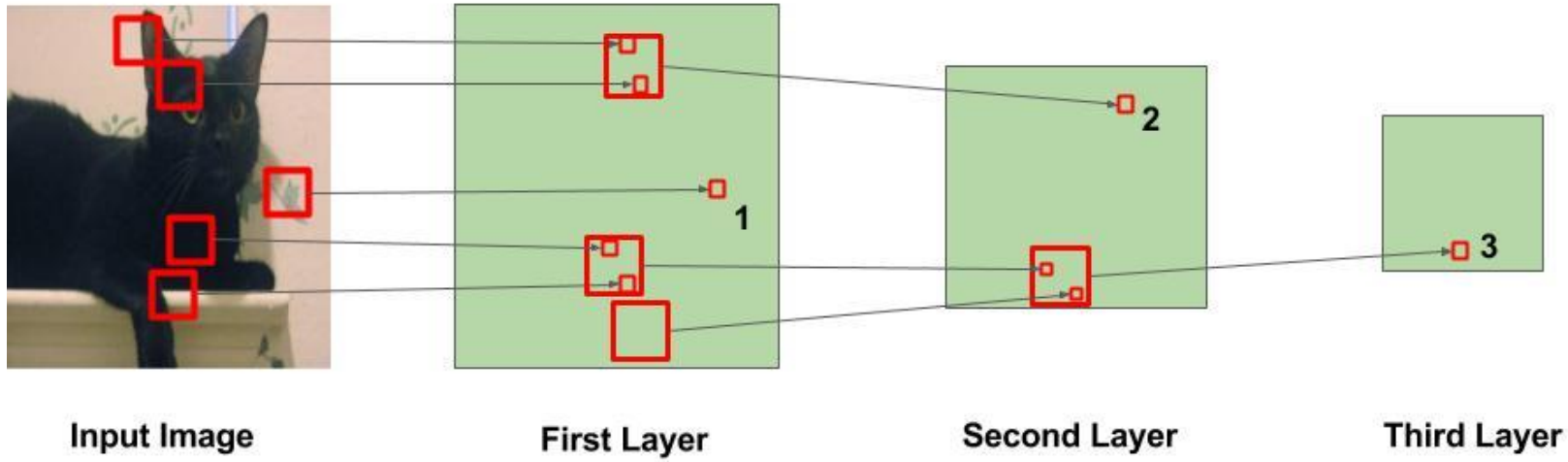
- 保持不變性 (invariance)
- 減少資訊量

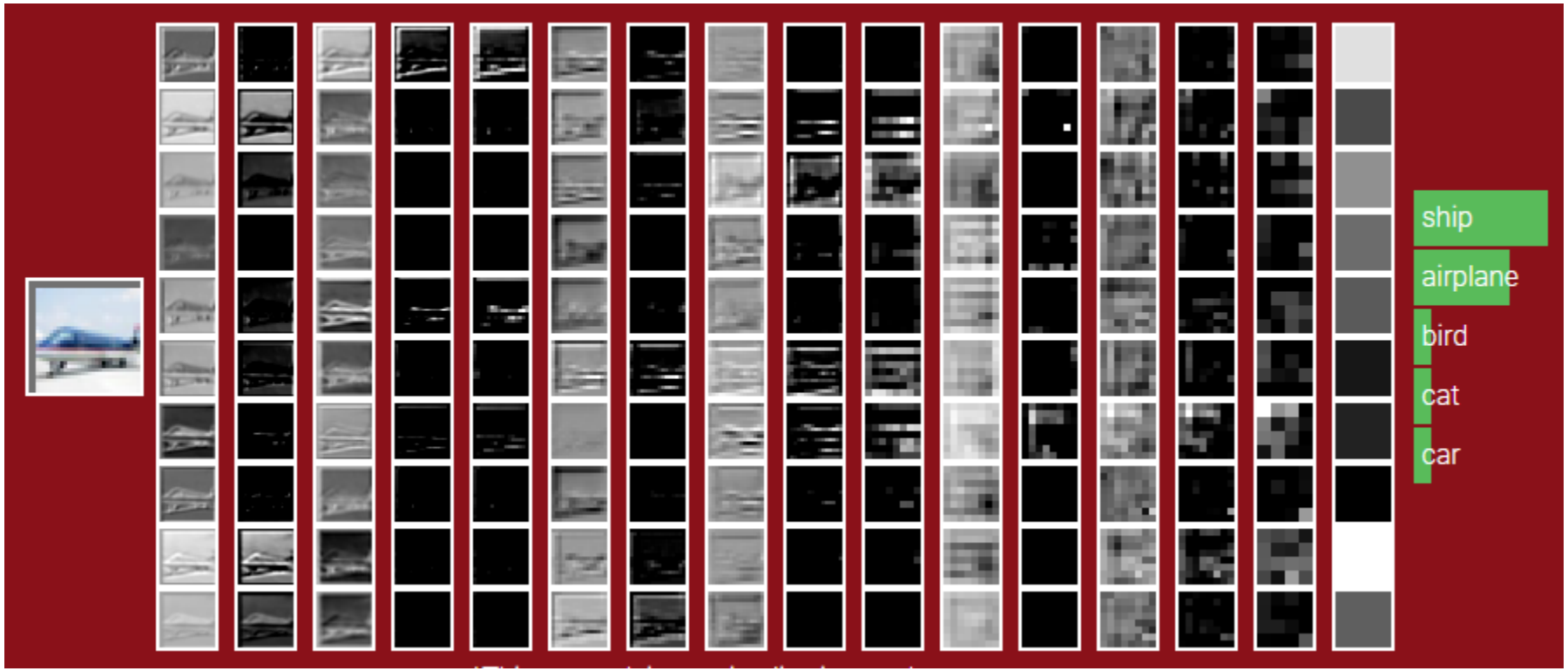
7	12	11	2
4	15	17	10
10	6	15	6
8	10	4	3

Max Pooling

15	17
10	15




1
2
3
Input Image
First Layer
Second Layer
Third Layer


ship
airplane
bird
cat
car

MNIST-CNN.ipynb

- **資料預處理：**
  - 因為要進行Convolution，需要調整input image的維度
  - 讓keras知道每一個filter有 “多厚”

```
train_images.shape
```

```
(60000, 28, 28)
```

```
train_labels.shape
```

```
(60000,)
```

```
# Translation of data
X_Train40 = train_images.reshape(train_images.shape[0], 28, 28, 1).astype('float32')

X_Train40.shape
```

```
(60000, 28, 28, 1)
```

```
y_Train.shape
```

```
(60000,)
```

- 建立模型：

```
# Convolution layer

model = Sequential()
# Create CN layer 1
model.add(Conv2D(filters=16, kernel_size=(5,5),
                 padding='same',input_shape=(28,28,1), activation=tf.nn.relu))
# Create Max-Pool 1
model.add(MaxPooling2D(pool_size=(2,2)))

# Create CN layer 2
model.add(Conv2D(filters=36, kernel_size=(5,5),
                 padding='same', activation=tf.nn.relu))

# Create Max-Pool 2
model.add(MaxPooling2D(pool_size=(2,2)))

# Add Dropout layer
model.add(Dropout(0.25))
```

```
# MLP
model.add(Flatten())
model.add(Dense(256, activation=tf.nn.relu, kernel_initializer='normal'))
model.add(Dropout(0.5))
model.add(Dense(10, activation=tf.nn.softmax, kernel_initializer='normal'))
```

```
# 定義訓練方式
model.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
model.summary()
```

- 訓練模型

```
# 開始訓練
train_history = model.fit(x=X_Train40_norm, y=y_Train, validation_split=0.2,
                          epochs=10, batch_size=300, verbose=2)
```

- 評估模型準確率

```
scores = model.evaluate(X_Test40_norm, y_Test)
print("\t[Info] Accuracy of testing data = {:2.1f}%".format(scores[1]*100.0))
```

- 預測

```
prediction = model.predict(X_Test40_norm)  # Making prediction and save result to prediction
print()
print("%s\n" % (prediction[0]))

print("%s\n" % (np.argmax(prediction[0])))
```

```
()
[1.1713490e-10 2.5775567e-08 5.3196395e-08 1.1177229e-06 1.3688411e-12
 5.7980698e-10 3.1437198e-15 9.9999809e-01 3.0639427e-09 8.8183458e-07]

7
```

• 預測位移的圖片

```
from util.my_plot import generate_shift_mnist_data
shift_img, shift_img_label = generate_shift_mnist_data(10000)
plot_batch(shift_img, shift_img_label,class_names ,5)
```


```
shift_img = shift_img.reshape(shift_img.shape[0], 28, 28, 1).astype('float32')
aa_norm = shift_img /255

scores = model.evaluate(aa_norm, shift_img_label)
print("Accuracy of generated shift data = {:2.1f}%".format(scores[1]*100.0))
```

```
10000/10000 [==============================] - 1s 147us/step
Accuracy of generated shift data = 72.2%
```

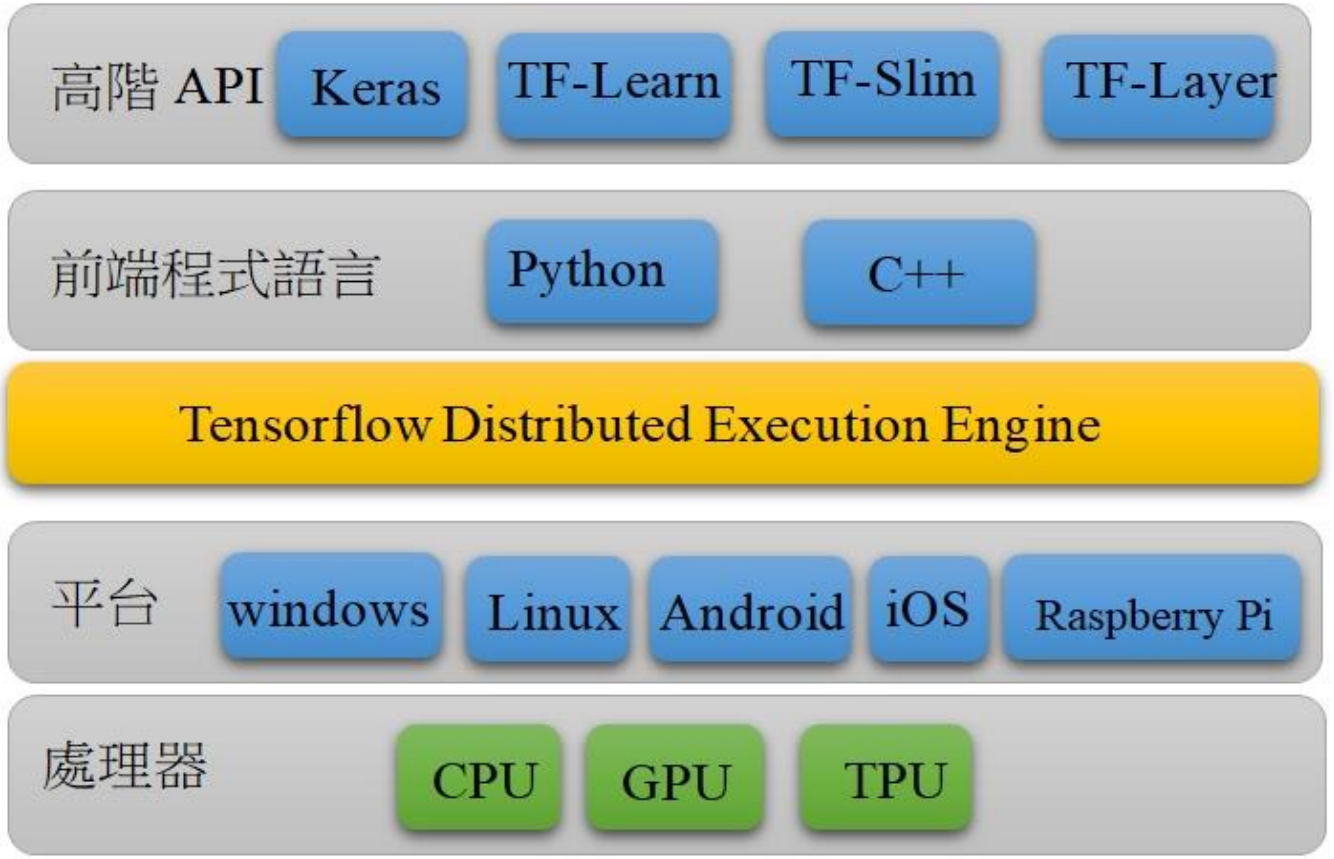
Exercise

課程大綱

- Python 基礎
  - Python、Numpy、Matplot、Panda
- 深度學習原理
  - Neural network 基礎
  - Keras v.s. Tensorflow
- Keras處理手寫數字辨識資料集介
  - 建立Multiple Layer perceptron model (MLP)
  - 建立 convolutional neural network (CNN)
- Tensorflow手寫數字辨識資料集介
  - 建立Multiple Layer perceptron model (MLP)
  - 建立 convolutional neural network (CNN)

TensorFlow – 架構

Tensorflow 架構圖說明:



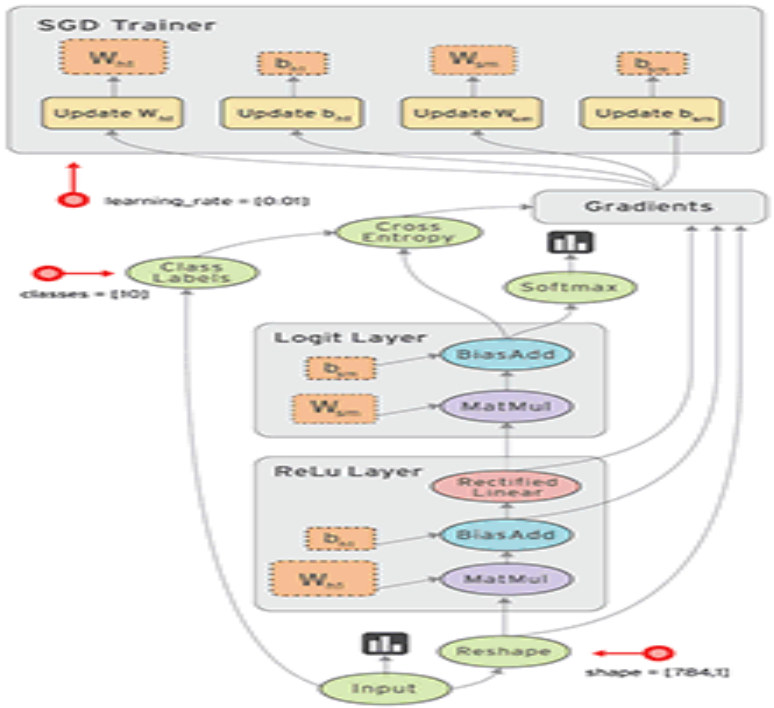
TensorFlow – 簡介

- TensorFlow是由Google 提供的開放原始碼程式庫。
- 組成：
  - Tensor：張量
  - Flow：資料流程

```
0.2
```
0 維的張量(數值)

```
[ 0.40000001  0.2         0.40000001]
```
1 維的張量(向量)

```
[[-0.5        -0.2         0.30000001]
 [-0.30000001  0.40000001  0.2       ]
 [ 0.80000001  0.2        -0.2       ]]
```
2 維以上的張量(矩陣)



TensorFlow – 概念

- 概念：
  - 用TensorFlow所提供的模組撰寫出『計算圖』
  - 在不同平台上，運算所定義好的『計算圖』

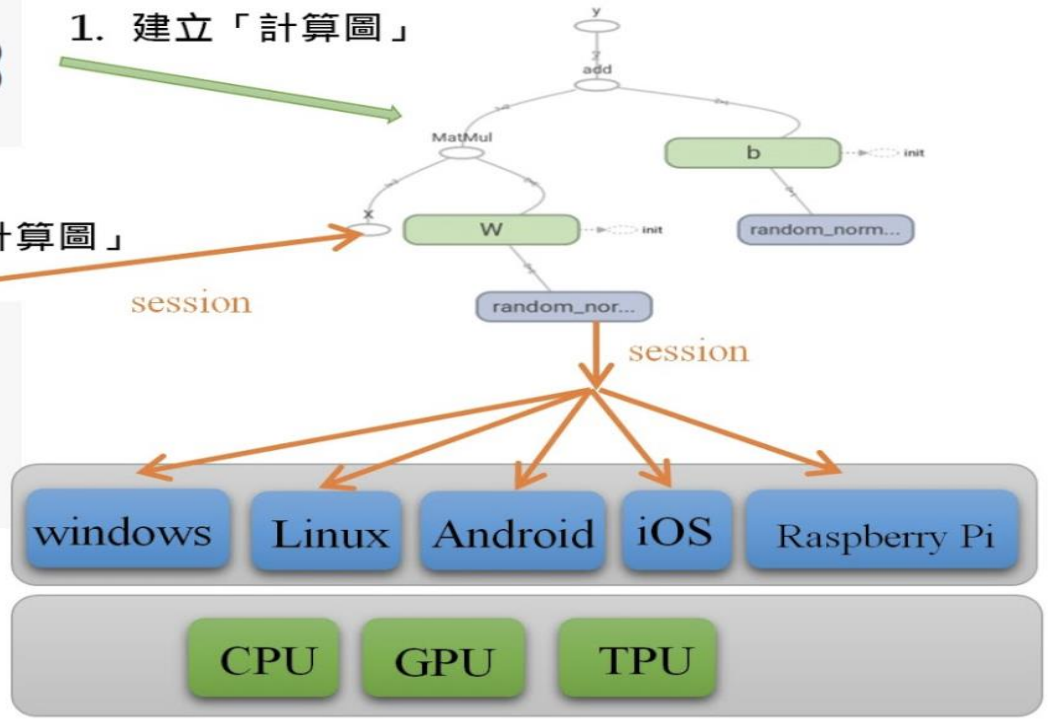
- y=activation(MathMul(X,W) + b)
  - X, W, b都是Tensor
  - W與b透過隨機化過程初始
  - X執行時代入

TensorFlow – 程式設計模式

- 程式設計模式：
  - 建立『計算圖』：
  - 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')
```

```
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})
```



TensorFlow – TF-Basic.ipynb

- 建立『計算圖』：(Lab - TensorFlow_Basic.ipynb)
  - Step 1. 匯入TensorFlow模組
  - Step 2. 建立TensorFlow常數
  - Step 3. 查看TensorFlow常數

建立 const

In [3]:
```
#建立TensorFlow常數
ts_c = tf.constant(2,name='ts_c')
```

In [4]:
```
#查看TensorFlow常數
ts_c
```

Out[4]:
```
<tf.Tensor 'ts_c:0' shape=() dtype=int32>
```

TensorFlow變數

- 建立『計算圖』：(Lab - TensorFlow_Basic.ipynb)
  - Step 4. 建立TensorFlow變數
  - Step 5. 查看TensorFlow變數

建立 Variable

```
In [5]: #建立TensorFlow變數
        ts_x = tf.Variable(ts_c+5,name='ts_x')

In [6]: #查看TensorFlow變數
        ts_x

Out[6]: <tf.Variable 'ts_x:0' shape=() dtype=int32_ref>
```

TensorFlow常數

- 執行『計算圖』：(Lab - TensorFlow_Basic.ipynb)
  - Step 1. 建立session
  - Step 2. 執行TensorFlow起始化變數
  - Step 3. 使用sess.run顯示TensorFlow常數
  - Step 4. 使用sess.run顯示TensorFlow變數

```
In [7]: #建立session
        sess=tf.Session()

In [8]: #執行TensorFlow起始化變數
        init = tf.global_variables_initializer()
        sess.run(init)

In [9]: #使用sess.run顯示TensorFlow常數
        print('ts_c=',sess.run(ts_c))

        ts_c= 2

In [10]: #使用sess.run顯示TensorFlow變數
         print('ts_x=',sess.run(ts_x))

         ts_x= 7
```

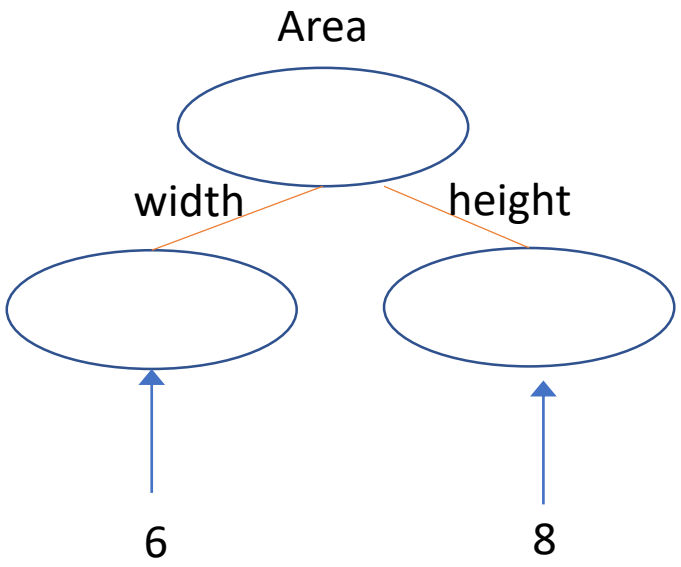
Placeholder

- 執行『計算圖』才設定數值
  - Step1. 建立計算圖
    - 給兩個placeholder分別為長與寬
  - Step2. 執行計算圖
    - 透過feed_dict給placeholder值

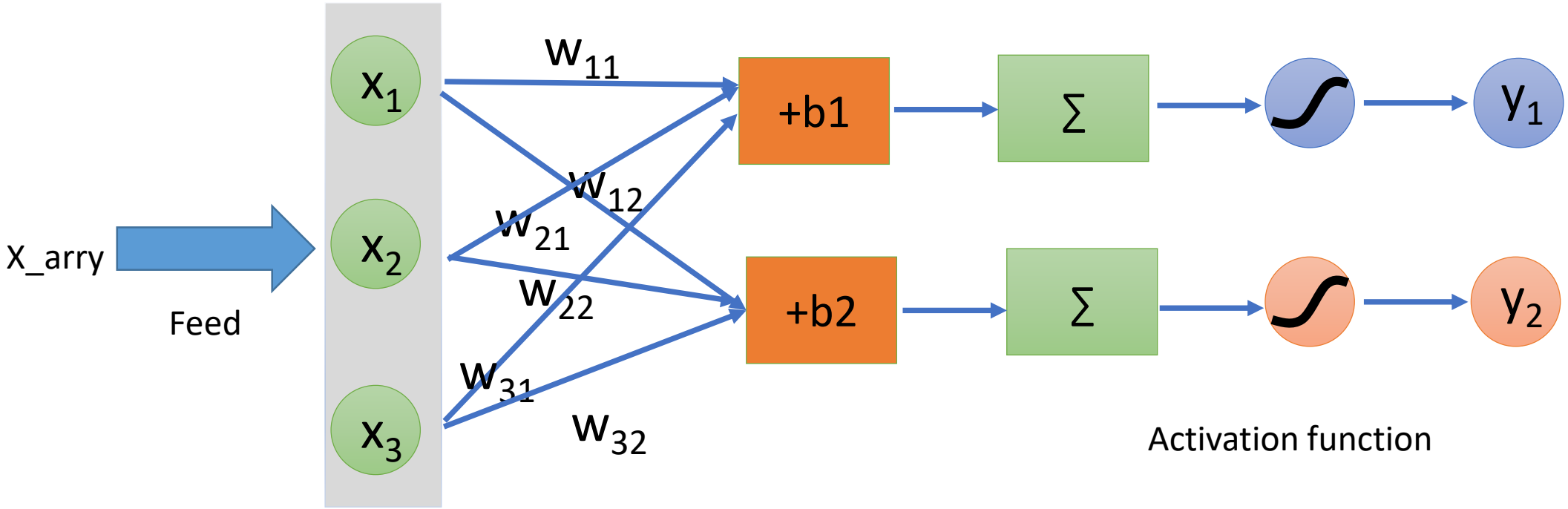
```
width = tf.placeholder("int32")
height = tf.placeholder("int32")
area=tf.multiply(width,height)
```

```
with tf.Session() as sess:
    init = tf.global_variables_initializer()
    sess.run(init)
    print('area=',sess.run(area,feed_dict={width: 6, height: 8}))
```

```
('area=', 48)
```



矩陣運算模擬神經網路


X_arry
Feed
x1
x2
x3
Place Holder
w11
w12
w21
w22
w31
w32
+b1
+b2
∑
∑
y1
y2
Activation function

y= relu ((X.W) + b)

In [2]:
```
X = tf.Variable([[0.4,0.2,0.4]])

W = tf.Variable([[-0.5,-0.2 ],
                 [-0.3, 0.4 ],
                 [-0.5, 0.2 ]])

b = tf.Variable([[0.1,0.2]])

XWb =tf.matmul(X,W)+b

y=tf.nn.relu(tf.matmul(X,W)+b)

with tf.Session() as sess:
    init = tf.global_variables_initializer()
    sess.run(init)
    print('XWb:')
    print(sess.run(XWb ))
    print('y:')
    print(sess.run(y ))
```

以placeholder傳入x值

```
W = tf.Variable(tf.random_normal([3, 2]))
b = tf.Variable(tf.random_normal([1, 2]))
X = tf.placeholder("float", [None,3])
y=tf.nn.relu(tf.matmul(X,W)+b)
```

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

    print('b:')
    print(_b)
    print('W:')
    print(_W)
    print('X:')
    print(_X)
    print('y:')
    print(_y)
```

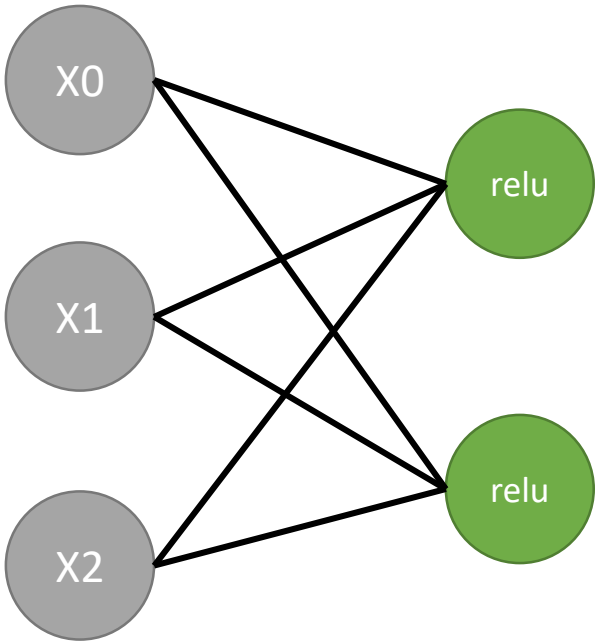
建立layer函數

- Layer函數：以矩陣運算模擬神經網路

```
def layer(output_dim,input_dim,inputs, activation=None):
    W = tf.Variable(tf.random_normal([input_dim, output_dim]))
    b = tf.Variable(tf.random_normal([1, output_dim]))
    XWb = tf.matmul(inputs, W) + b
    if activation is None:
        outputs = XWb
    else:
        outputs = activation(XWb)
    return outputs
```

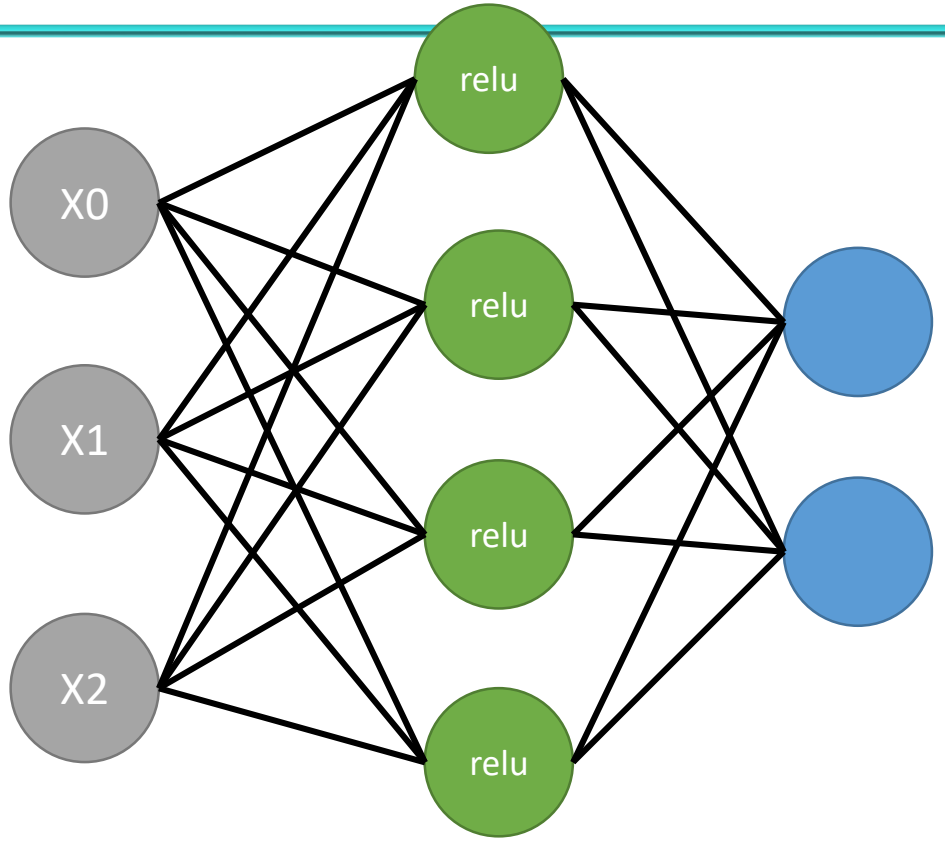
一層的MLP

```
X = tf.placeholder("float", [None,3])

y=layer(output_dim=2,input_dim=3,inputs=X, activation=tf.nn.relu)

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]])
    (_X,_y)=sess.run((X,y),feed_dict={X:X_array})
    print('X:')
    print(_X)
    print('y:')
    print(_y)
    tf.summary.merge_all()
    train_writer = tf.summary.FileWriter('log/area',sess.graph)
```



二層的MLP

```
X = tf.placeholder("float", [None,3])
h=layer(output_dim=4,input_dim=3,inputs=X,
        activation=tf.nn.relu)
y=layer(output_dim=2,input_dim=4,inputs=h)
with tf.Session() as sess:
    init = tf.global_variables_initializer()
    sess.run(init)
    X_array = np.array([[0.4,0.2 ,0.4]])
    (layer_X,layer_h,layer_y)= sess.run((X,h,y),feed_dict={X:X_array})
    print('input Layer X:')
    print(layer_X)
    print('hidden Layer h:')
    print(layer_h)
    print('output Layer y:')
    print(layer_y)
```



課程大綱

- Python 基礎
  - Python、Numpy、Matplot、Panda
- 深度學習原理
  - Neural network 基礎
  - Keras v.s. Tensorflow
- Keras處理手寫數字辨識資料集介
  - 建立Multiple Layer perceptron model (MLP)
  - 建立 convolutional neural network (CNN)
- Tensorflow手寫數字辨識資料集介
  - 建立Multiple Layer perceptron model (MLP)
  - 建立 convolutional neural network (CNN)

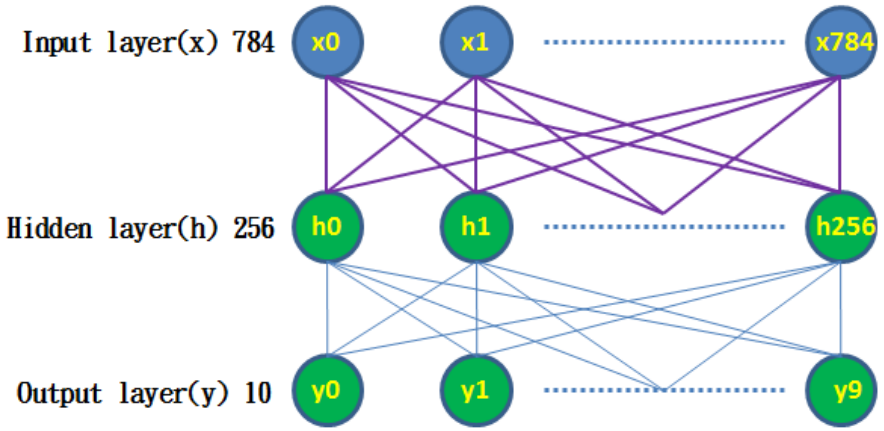
設計graph: 建立模型

```
# 建立輸入層 x
x = tf.placeholder("float", [None, 784])
```

```
# 建立隱藏層h1
D_Hidden=layer(output_dim=256,input_dim=784, inputs=x ,activation=tf.nn.relu)
```

```
# 建立輸出層
y_predict=layer(output_dim=10,input_dim=256, inputs=D_Hidden_Dropout,activation=tf.nn.softmax)
```

做預測用的graph




```
model = Sequential()
model.add(Flatten(input_shape=(28, 28)))
model.add(Dense(256, activation=tf.nn.relu, kernel_initializer='normal'))
model.add(Dense(10, activation=tf.nn.softmax, kernel_initializer='normal'))
```

設計graph: 訓練 & 評估

```
#定義loss function

loss_function = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(logits=y_predict, labels=y_label))
```

評估用的graph

```
#選擇optimizer

optimizer = tf.train.AdamOptimizer(learning_rate=0.001).minimize(loss_function)
```

訓練用的graph

```
#計算每一筆資料是否正確預測

correct_prediction = tf.equal(tf.argmax(y_label   , 1), tf.argmax(y_predict, 1))

#將計算預測正確結果，加總平均

accuracy = tf.reduce_mean(tf.cast(correct_prediction, "float"))
```

評估用的graph


```
model.compile(optimizer=tf.train.AdamOptimizer(),
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

執行graph: 訓練模型

```
# 進行多次實驗
for epoch in range(trainEpochs):
    # 每一步取一批做運算
    for i in range(totalBatchs):
        batch_x, batch_y = mnist.train.next_batch(batchSize)
        sess.run(optimizer,feed_dict={x: batch_x/255 ,y_label: batch_y})

    # 計算training data的準確度
    loss,acc = sess.run([loss_function,accuracy],
                        feed_dict={x: mnist.train.images/255,
                                   y_label: mnist.train.labels})


    # 計算validate data的準確度
    val_loss,val_acc = sess.run([loss_function,accuracy],
                                feed_dict={x: mnist.validation.images/255,
                                           y_label: mnist.validation.labels})


    epoch_list.append(epoch)
    loss_list.append(loss);accuracy_list.append(acc)
    val_loss_list.append(val_loss);val_accuracy_list.append(val_acc)
    print("Train Epoch:", '%02d' % (epoch+1), "Loss=", "{:.9f}".format(loss), " Accuracy=",acc ,"val Loss=", "{:.9f}".format(val_loss)," val_A

duration =time()-startTime
print("Train Finished takes:",duration)
```

```
train_history = model.fit(train_images, train_labels, validation_split=0.2, epochs=10, batch_size=200)

Train on 48000 samples, validate on 12000 samples
Epoch 1/10
48000/48000 [==============================] - 1s 17us/step - loss: 0.0813 - acc: 0.9749 - val_loss: 0.0788 - val_acc: 0.9777
Epoch 2/10
48000/48000 [==============================] - 1s 17us/step - loss: 0.0773 - acc: 0.9764 - val_loss: 0.0774 - val_acc: 0.9773
Epoch 3/10
48000/48000 [==============================] - 1s 17us/step - loss: 0.0746 - acc: 0.9764 - val_loss: 0.0751 - val_acc: 0.9788
```

設計graph: 評估模型

```
print("Accuracy:", sess.run(accuracy,
                            feed_dict={x: mnist.test.images/255,
                                       y_label: mnist.test.labels}))
```

```
('Accuracy:', 0.5466)
```


```
scores = model.evaluate(test_images, test_labels)
print("Accuracy of testing data = {:2.1f}%".format(scores[1]*100.0))
```

```
10000/10000 [==============================] - 0s 14us/step
Accuracy of testing data = 98.1%
```

設計graph: 進行預測

```
prediction_result=sess.run(tf.argmax(y_predict,1),
                           feed_dict={x: mnist.test.images/255 })
```

```
prediction_result[:10]
```

```
array([3, 3, 3, 0, 4, 3, 4, 9, 5, 9])
```

```
predictions = model.predict(test_images)
print predictions.shape
print("\n")
print predictions[0]
```

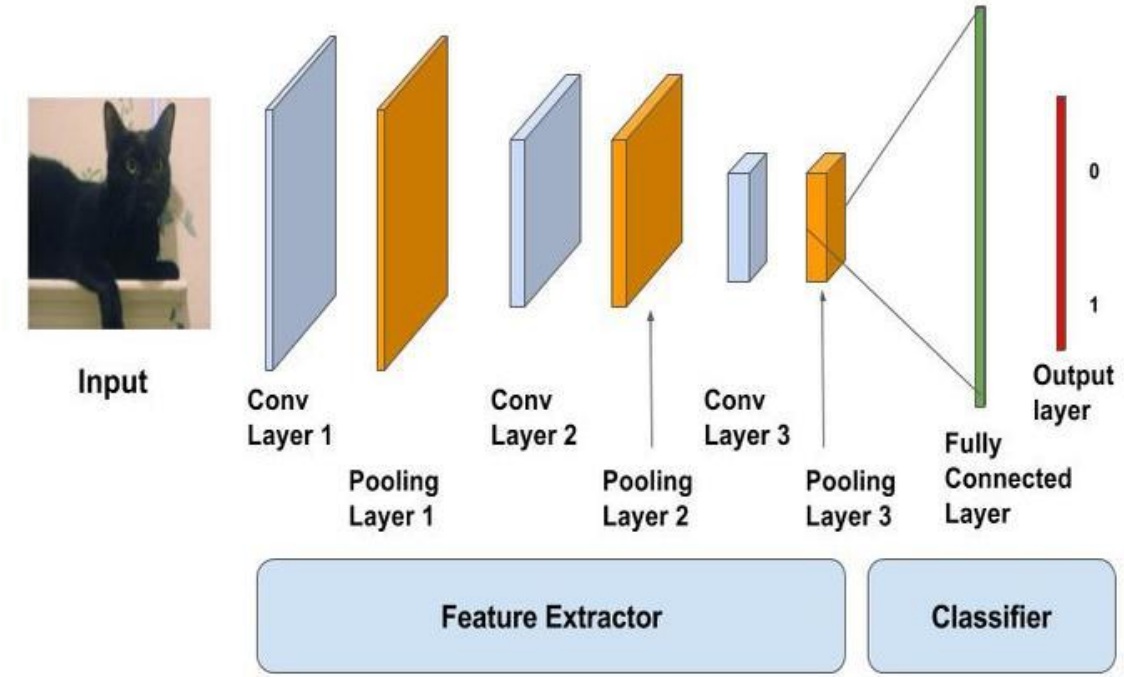
```
(10000, 10)


[5.5651244e-09 1.5294382e-10 4.6140903e-07 5.0855342e-05 4.9134763e-15
 4.5012856e-09 3.9966551e-15 9.9994826e-01 9.3465147e-09 3.4660687e-07]
```

課程大綱

- Python 基礎
  - Python、Numpy、Matplot、Panda
- 深度學習原理
  - Neural network 基礎
  - Keras v.s. Tensorflow
- Keras處理手寫數字辨識資料集介
  - 建立Multiple Layer perceptron model (MLP)
  - 建立 convolutional neural network (CNN)
- Tensorflow手寫數字辨識資料集介
  - 建立Multiple Layer perceptron model (MLP)
  - 建立 convolutional neural network (CNN)

Tensorflow - CNN

- 取特徵後，再丟給MLP
  - Convolution Layer
  - Max Pooling Layer



建立graph: 共用函數

- 產生kernel filter

```
def kernel(shape):
    return tf.Variable(tf.truncated_normal(shape, stddev=0.1))
```

- 執行convolution

```
def conv2d(x, W):
    return tf.nn.conv2d(x, W, strides=[1,1,1,1], padding='SAME')
```

- 執行max pooling

```
def max_pool_2x2(x):
    return tf.nn.max_pool(x, ksize=[1,2,2,1], strides=[1,2,2,1], padding='SAME')
```

建立graph:建立模型

```
x = tf.placeholder("float", [None, 784])
x_image = tf.reshape(x, [-1, 28, 28, 1])
```

```
# convolution
W1 = kernel([ 5,5,1,16])
Conv1=conv2d(x_image, W1)
C1_Conv = tf.nn.relu(Conv1)

# max pool
C1_Pool = max_pool_2x2(C1_Conv)
```

```
# convolution
W2 = kernel([5,5,16,36])
Conv2=conv2d(C1_Pool, W2)
C2_Conv = tf.nn.relu(Conv2)

# max pool
C2_Pool = max_pool_2x2(C2_Conv)
```

```
D_Flat = tf.reshape(C2_Pool, [-1, 1764])
```

```
# 建立輸入層 x
x = tf.placeholder("float", [None, 784])
```

```
# 建立隱藏層h1
D_Hidden=layer(output_dim=256,input_dim=784, inputs=x ,activation=tf.nn.relu)
```

```
# 建立輸出層
y_predict=layer(output_dim=10,input_dim=256, inputs=D_Hidden_Dropout,activation=tf.nn.softmax)
```

```
D_Hidden=layer(output_dim=256,input_dim=1764, inputs=D_Flat ,activation=tf.nn.relu)
y_predict=layer(output_dim=10,input_dim=256, inputs=D_Hidden,activation=tf.nn.softmax)
```

設計graph: 訓練 & 評估

```
#定義loss function

loss_function = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(logits=y_predict, labels=y_label))
```

評估用的graph

```
#選擇optimizer

optimizer = tf.train.AdamOptimizer(learning_rate=0.001).minimize(loss_function)
```

訓練用的graph

```
#計算每一筆資料是否正確預測

correct_prediction = tf.equal(tf.argmax(y_label   , 1), tf.argmax(y_predict, 1))

#將計算預測正確結果，加總平均

accuracy = tf.reduce_mean(tf.cast(correct_prediction, "float"))
```

評估用的graph

執行graph: 訓練模型

```
# 進行多次實驗
for epoch in range(trainEpochs):
    # 每一步取一批做運算
    for i in range(totalBatchs):
        batch_x, batch_y = mnist.train.next_batch(batchSize)
        sess.run(optimizer, feed_dict={x: batch_x/255 ,y_label: batch_y})

    # 計算training data的準確度
    loss,acc = sess.run([loss_function,accuracy],
                        feed_dict={x: mnist.train.images[:5000]/255,
                                   y_label: mnist.train.labels[:5000]})


    # 計算validate data的準確度
    val_loss,val_acc = sess.run([loss_function,accuracy],
                                feed_dict={x: mnist.validation.images/255,
                                           y_label: mnist.validation.labels})


    epoch_list.append(epoch)
    loss_list.append(loss);accuracy_list.append(acc)
    val_loss_list.append(val_loss);val_accuracy_list.append(val_acc)
    print("Train Epoch:", '%02d' % (epoch+1), "Loss=", "{:.9f}".format(loss), " Accuracy=",acc ,"val Loss=", "{:.9f}".format(val_loss)," val_A

duration =time()-startTime
print("Train Finished takes:",duration)
```

設計graph: 評估模型

```
print("Accuracy:", sess.run(accuracy,
                            feed_dict={x: mnist.test.images/255,
                                       y_label: mnist.test.labels}))
```

```
('Accuracy:', 0.5466)
```

設計graph: 進行預測

```
prediction_result=sess.run(tf.argmax(y_predict,1),
                           feed_dict={x: mnist.test.images/255 })
```

```
prediction_result[:10]
```

```
array([3, 3, 3, 0, 4, 3, 4, 9, 5, 9])
```

Thank You!!