“Hello world” of deep learning

Keras

If you want to learn theano:

http://speech.ee.ntu.edu.tw/~tlkagk/courses/MLDS_2015_2/Lecture/Theano%20DNN.ecm.mp4/index.html

http://speech.ee.ntu.edu.tw/~tlkagk/courses/MLDS_2015_2/Lecture/RNN%20training%20(v6).ecm.mp4/index.html

TensorFlow or theano

Very flexible

Need some effort to learn

Interface of TensorFlow or Theano

keras

Easy to learn and use

(still have some flexibility)

You can modify it if you can write TensorFlow or Theano

Keras

- François Chollet is the author of Keras.
  - He currently works for Google as a deep learning engineer and researcher.

牛角

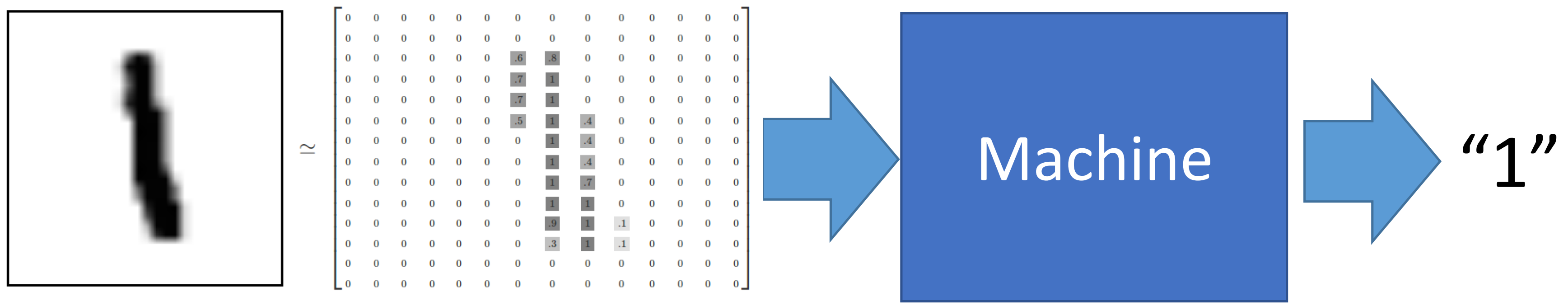
- Keras means *horn* in Greek
- Documentation: http://keras.io/
- Example: https://github.com/fchollet/keras/tree/master/examples

感謝 沈昇勳 同學提供圖檔

使用 Keras 心得

“Hello world”

- Handwriting Digit Recognition



$$\cong \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & .6 & .8 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & .7 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & .7 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & .5 & 1 & .4 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & .4 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & .4 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & .7 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & .9 & 1 & .1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & .3 & 1 & .1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

28 x 28

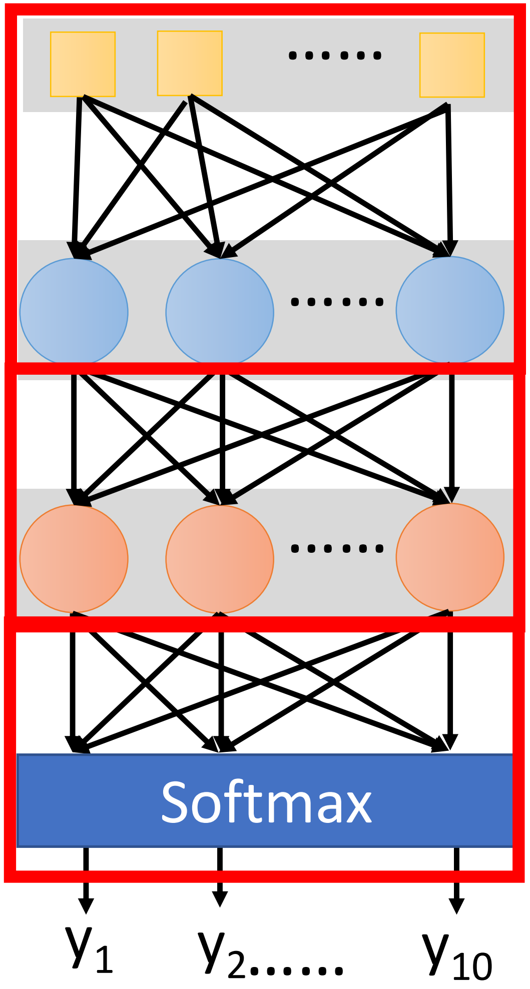
MNIST Data: http://yann.lecun.com/exdb/mnist/

Keras provides data sets loading function: http://keras.io/datasets/

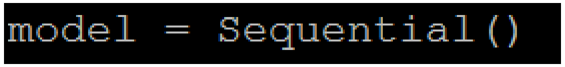
Keras

先宣告model(nn要長什麼樣子)

Step 1: define a set of function → **Step 2: goodness of function** → **Step 3: pick the best function**



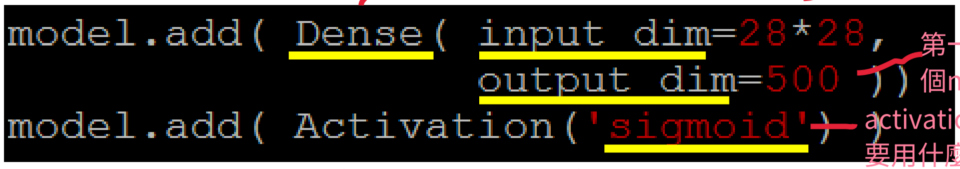
2個hidden layers 各500個neuron



```
model = Sequential()
```

dense=fully connected的layer dim=dimension

imput28*28的vector 代表image

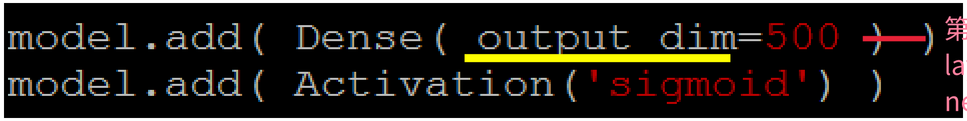


```
model.add( Dense( input_dim=28*28,
                  output_dim=500 ) )
model.add( Activation('sigmoid') )
```

第一個layer500個neuron

activation function 要用什麼

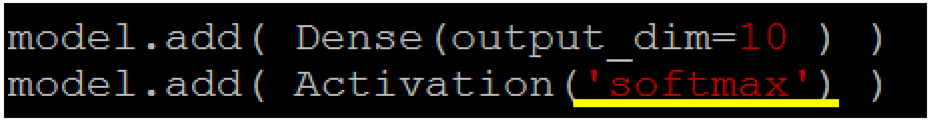
別的functions: softplus, softsign, relu, tanh, hard_sigmoid, linear



```
model.add( Dense( output_dim=500 ) )
model.add( Activation('sigmoid') )
```

第二個layer500個neuron

(不需要再寫input，因為第一個layer的output就是input)



```
model.add( Dense(output_dim=10 ) )
model.add( Activation('softmax') )
```

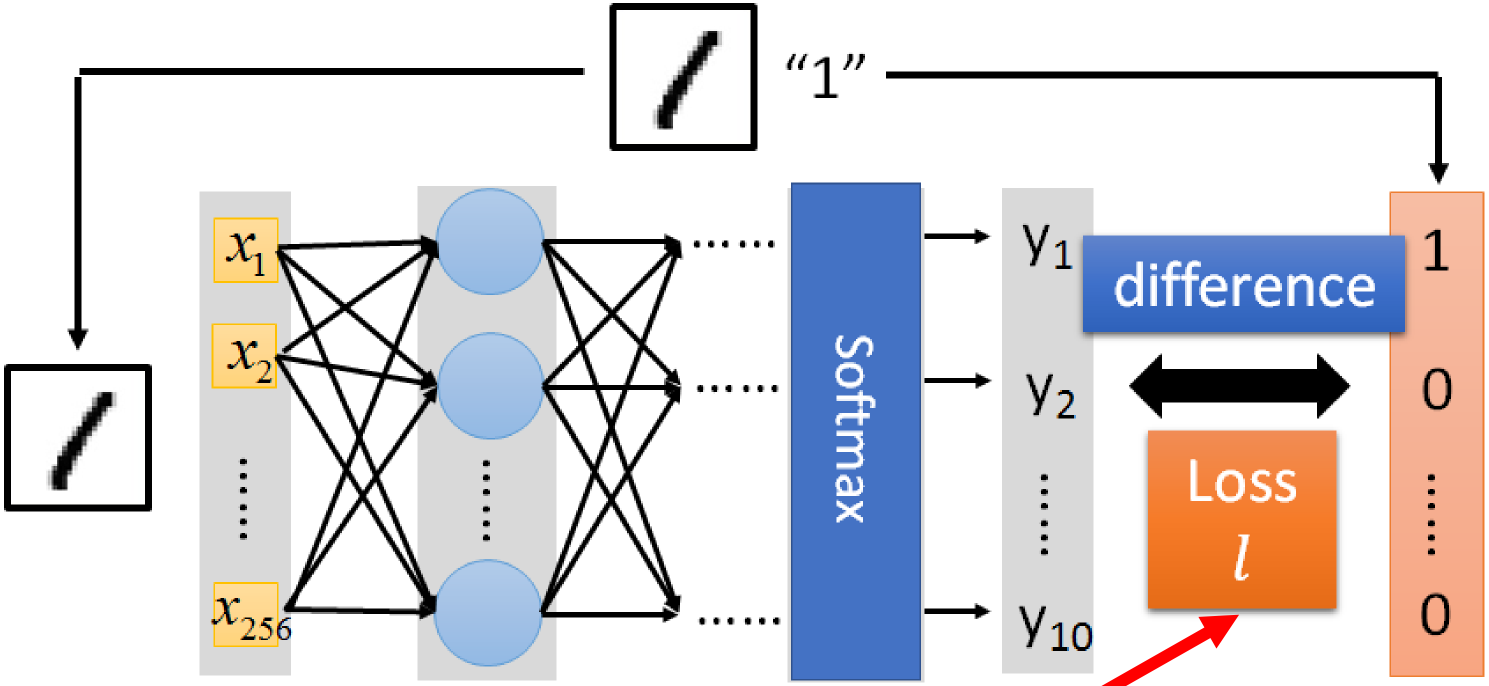
10維

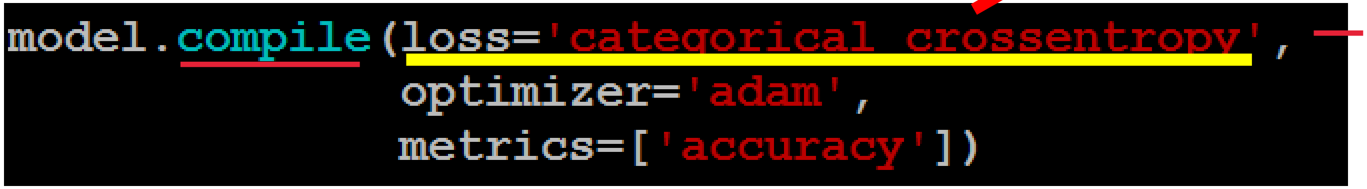
multi-class classfier

Keras

評估function的好壞

Step 1: define a set of function → **Step 2: goodness of function** → Step 3: pick the best function



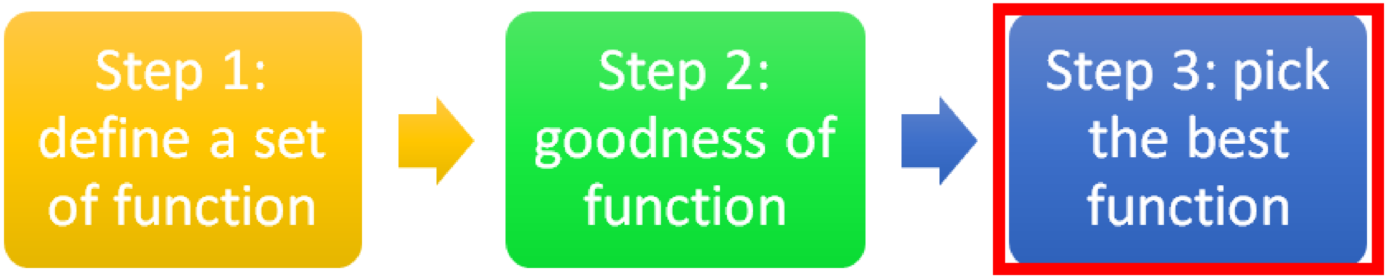


```
model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
```

定義loss function

Several alternatives: https://keras.io/objectives/

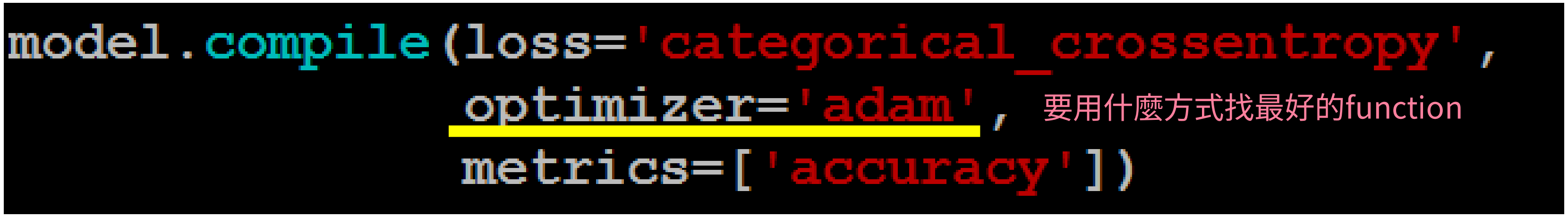
Keras


Step 1: define a set of function

Step 2: goodness of function

Step 3: pick the best function

Step 3.1: Configuration



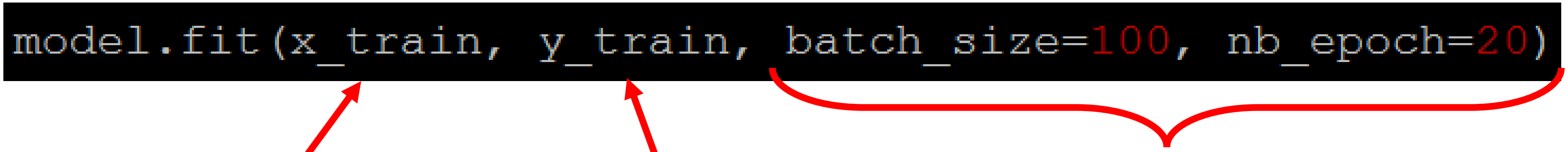
```
model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
```

要用什麼方式找最好的function

SGD, RMSprop, Adagrad, Adadelta, Adam, Adamax, Nadam

(通常這些方法原理都類似gradient desent，只是差別在learning rate是人自己設，還是機器自動設)

Step 3.2: Find the optimal network parameters



```
model.fit(x_train, y_train, batch_size=100, nb_epoch=20)
```

Training data (Images)

Labels (digits)

每一張image對應到0-9的哪個數字

In the following slides

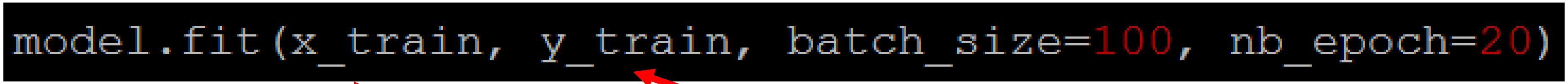
Keras

Step 1: define a set of function

Step 2: goodness of function

Step 3: pick the best function

Step 3.2: Find the optimal network parameters



```
model.fit(x_train, y_train, batch_size=100, nb_epoch=20)
```

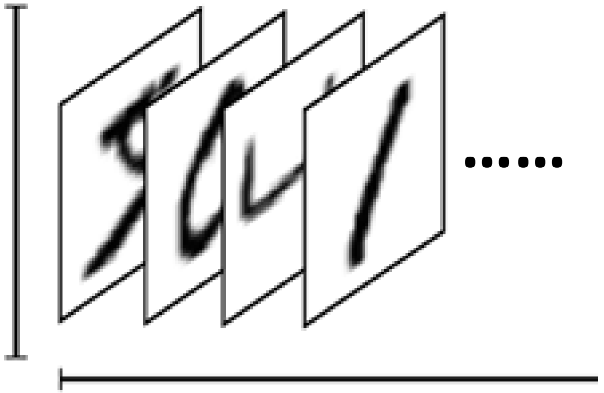
2 dimension matrix:
幾個example * image有多大

numpy array

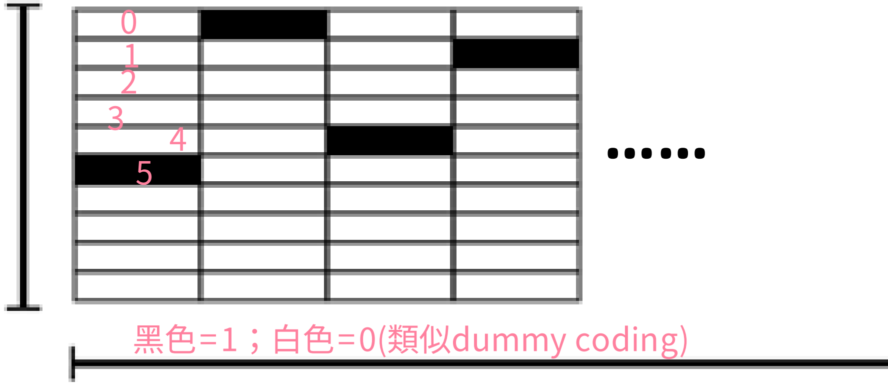
2 dimension matrix:
幾個trainiing example * output幾維

numpy array

28 x 28
=784
pixel



10



黑色=1；白色=0(類似dummy coding)

Number of training examples

每一個dimension是784維

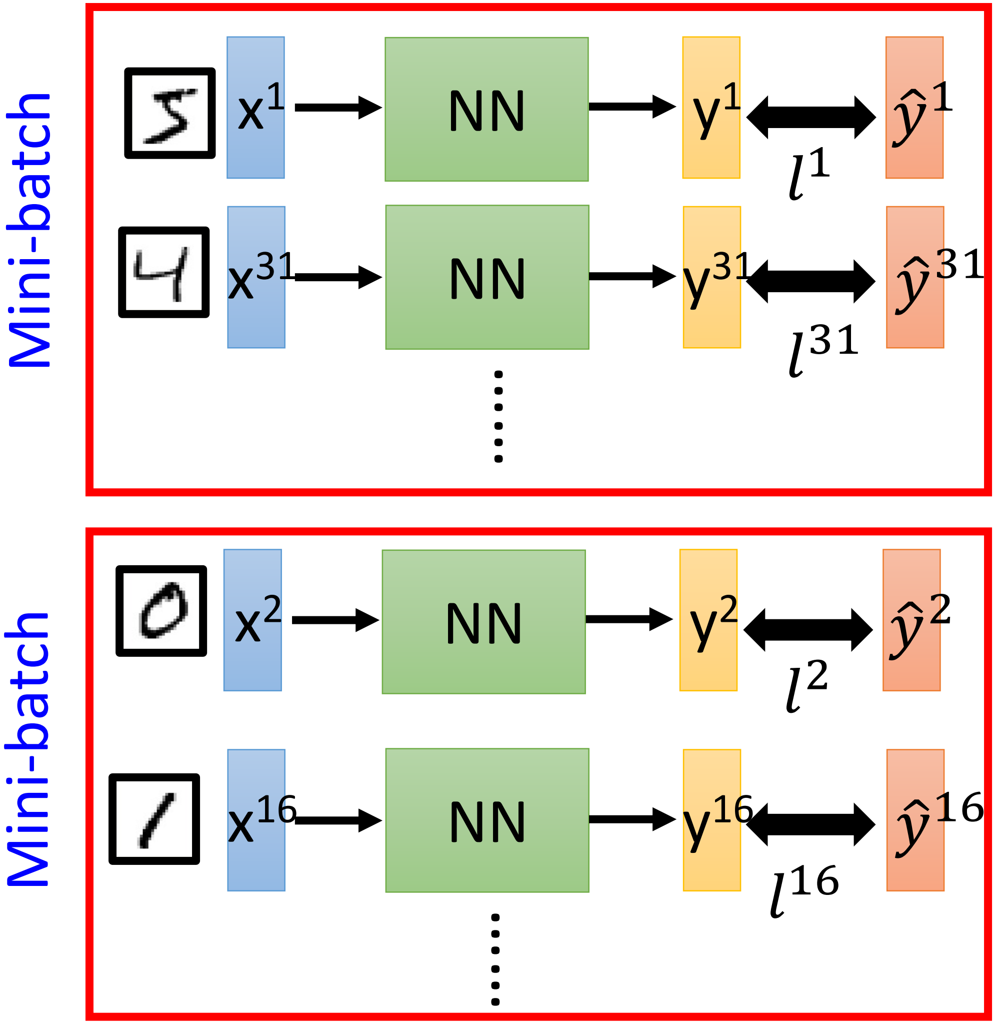
Number of training examples

https://www.tensorflow.org/versions/r0.8/tutorials/mnist/beginners/index.html

而是把training data隨機分成一個個的batch ←

We do not really minimize total loss!

Mini-batch



- ➢ Randomly initialize network parameters
- ➢ Pick the 1st batch

  隨機選一個batch出來

  $$L' = l^1 + l^{31} + \cdots$$

  計算第一個batch裡的element的total loss

  再根據L'更新參數

  Update parameters once
- ➢ Pick the 2nd batch

  $$L'' = l^2 + l^{16} + \cdots$$

  Update parameters once

  ⋮
- ➢ Until all mini-batches have been picked

  假設有100個batch就要做100次更新參數

one epoch

把所有batch看過一次=1個epoch

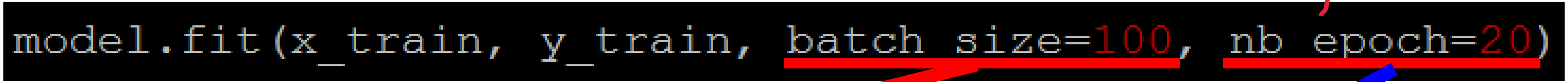
Repeat the above process

(通常做一個nn會需要好幾十個epoch)

Mini-batch

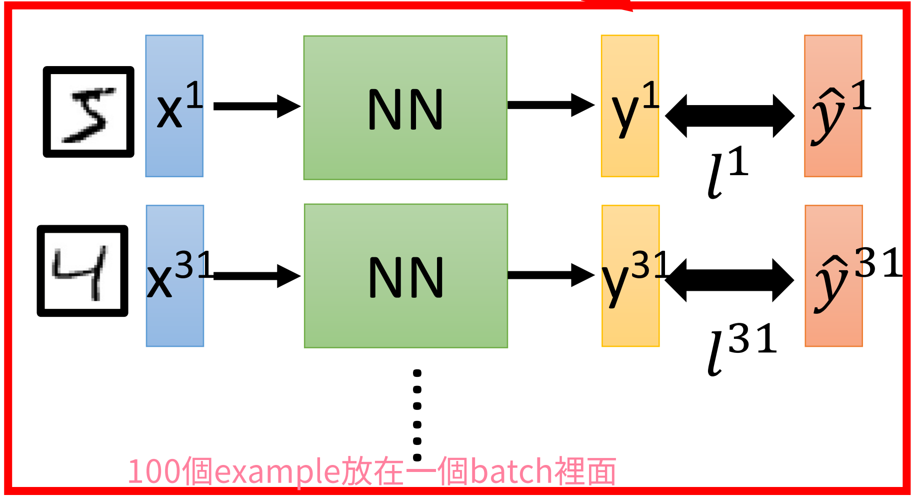
Batch size influences both ***speed*** and ***performance***. You have to tune it.

number of epoch



```
model.fit(x_train, y_train, batch_size=100, nb_epoch=20)
```

Mini-batch



100個example放在一個batch裡面

100 examples in a mini-batch

Batch size = 1 ➡ Stochastic gradient descent

相較於原本的gradient desent速度比較快

每個batch被看過20次，共更新20*100=2000次參數

- Pick the 1st batch

  $L' = l^1 + l^{31} + \cdots$

  Update parameters once

- Pick the 2nd batch

  $L'' = l^2 + l^{16} + \cdots$

  Update parameters once

  ⋮

- Until all mini-batches have been picked

Repeat 20 times

one epoch

定batch size: 實作上的issue，運算時間不同

Speed

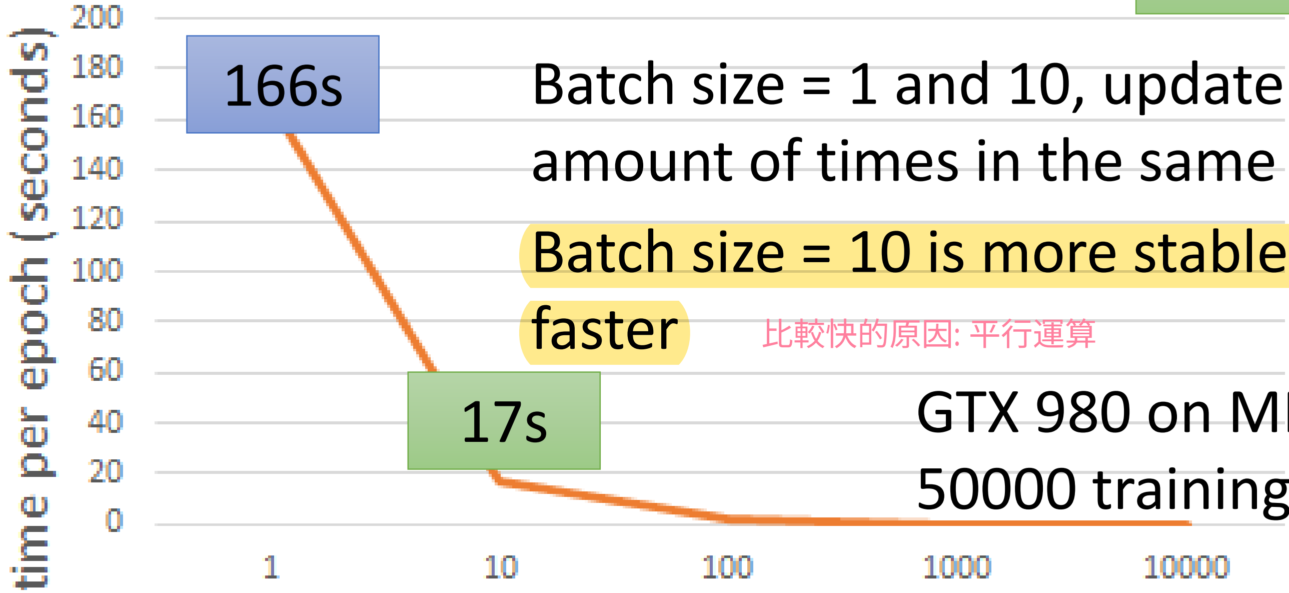
Very large batch size can yield worse performance

- Smaller batch size means more updates in one epoch
  - E.g. 50000 examples
  - batch size = 1, 50000 updates in one epoch
  - batch size = 10, 5000 updates in one epoch

(1) batch size=1, 一個epoch 更新50000次參數
(2) batch size=10, 一個epoch 更新5000次參數

166s 1 epoch

17s 10 epoch



Batch size = 1 and 10, update the same amount of times in the same period.

Batch size = 10 is more stable, converge faster

比較快的原因: 平行運算

GTX 980 on MNIST with 50000 training examples

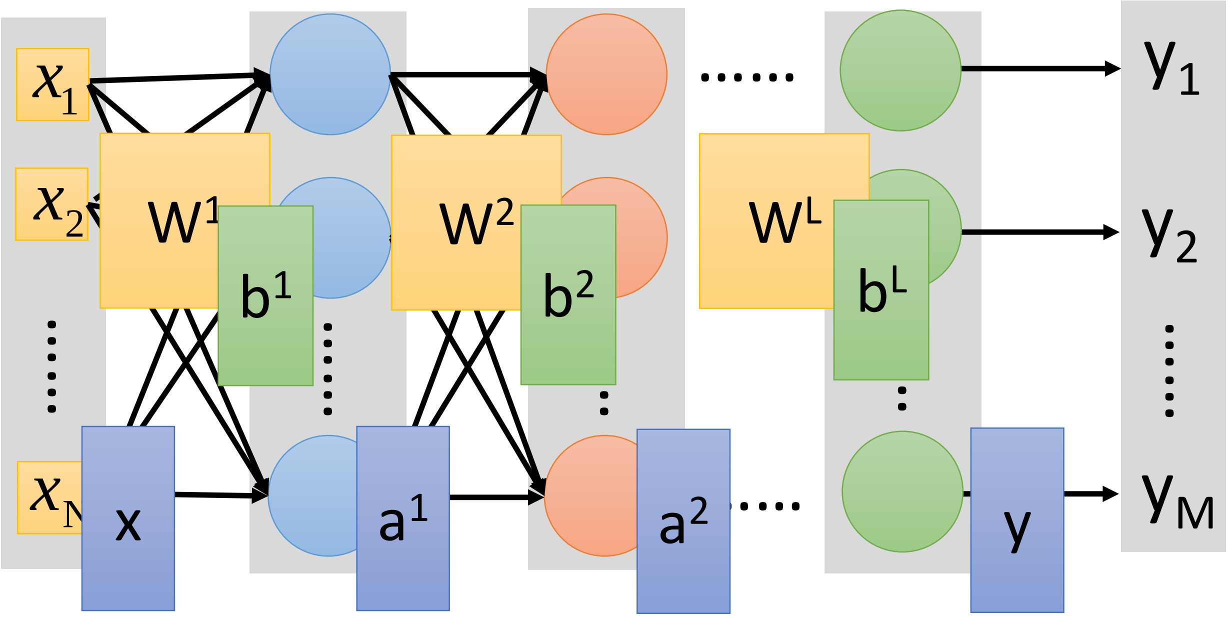
1 1000

所以size小做gradient descent時比較不會卡

隨機性 大 小

batch size開超大就好?
(1)硬體GPU無法平行運算
(2)train幾下就卡在local minimum或saddle point

Speed - Matrix Operation



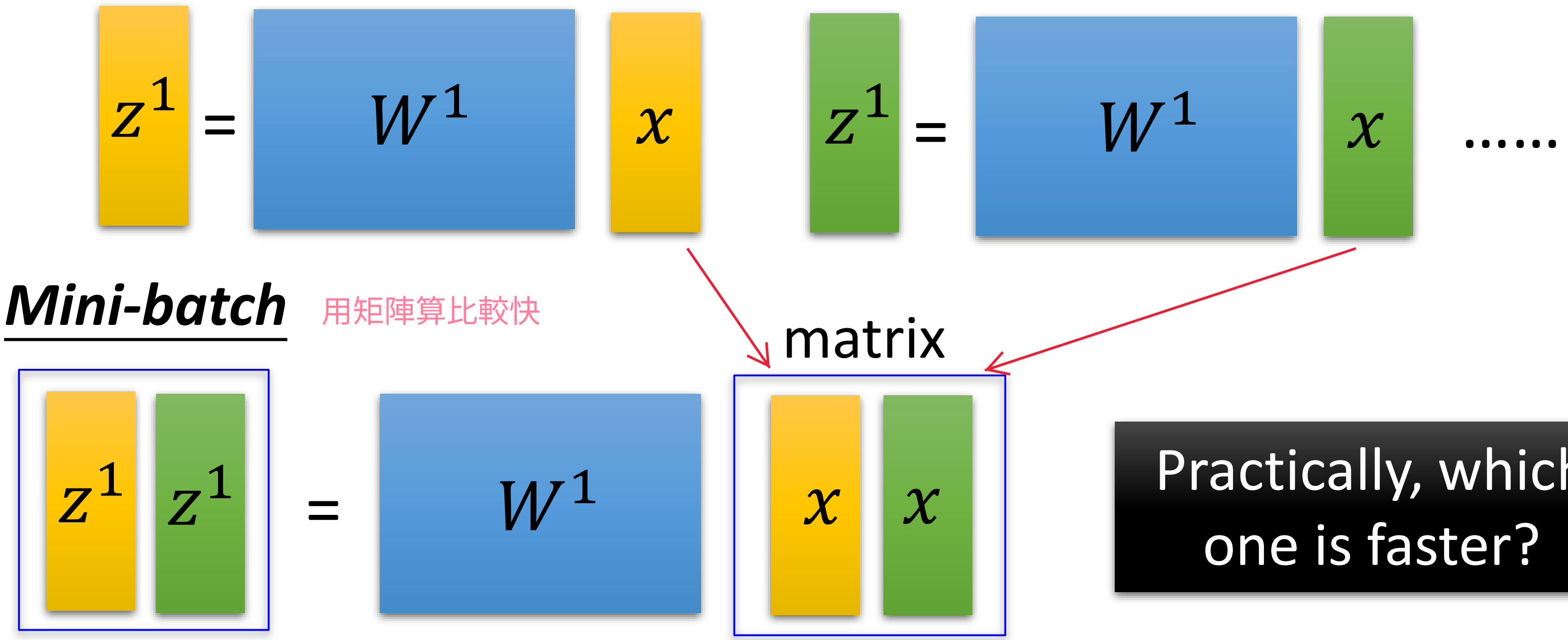
$$y = f(x)$$ Forward pass (Backward pass is similar)

$$= \sigma(W^L \cdots \sigma(W^2 \; \sigma(W^1 \; x + b^1) + b^2) \cdots + b^L)$$

Speed - Matrix Operation

- Why mini-batch is faster than stochastic gradient descent?

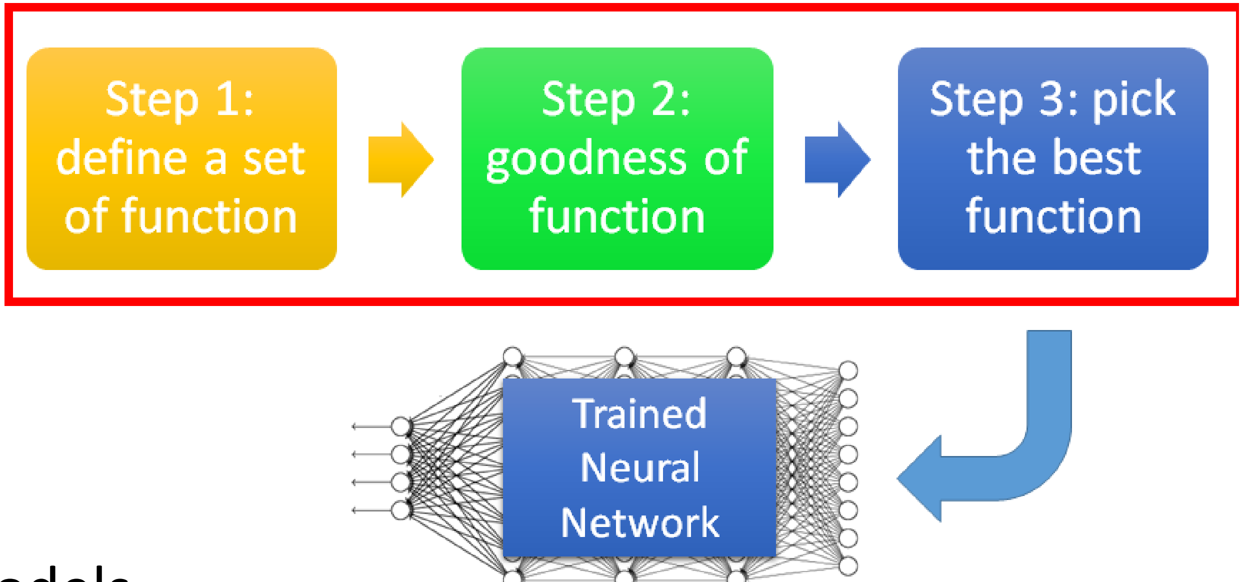
Stochastic Gradient Descent



Mini-batch 用矩陣算比較快

Practically, which one is faster?

Keras


Step 1: define a set of function
Step 2: goodness of function
Step 3: pick the best function
Trained Neural Network

Save and load models

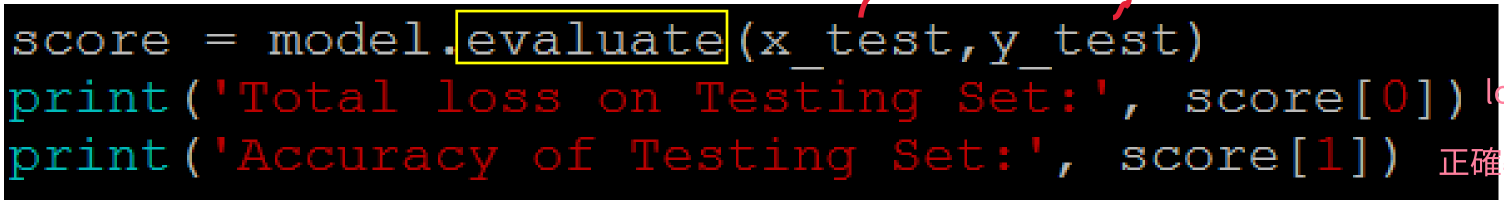
http://keras.io/getting-started/faq/#how-can-i-save-a-keras-model

How to use the neural network (testing):

testing image testing label ← input

評估model

case 1:



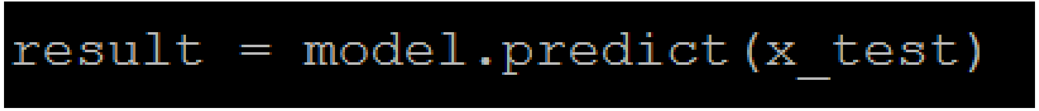
```
score = model.evaluate(x_test,y_test)
print('Total loss on Testing Set:', score[0])
print('Accuracy of Testing Set:', score[1])
```

loss

正確率

系統上線給使用者predict (所以只有X，output就是分類的結果y)

case 2:



```
result = model.predict(x_test)
```

Keras

- Using GPU to speed training
  - Way 1
    - THEANO_FLAGS=device=gpu0 python YourCode.py
  - Way 2 (in your code)
    - import os
    - os.environ["THEANO_FLAGS"] = "device=gpu0"

Live Demo