Python for Data Analysis

類神經網路

講者:楊翔斌
n07061033@mail.ncku.edu.tw

大綱

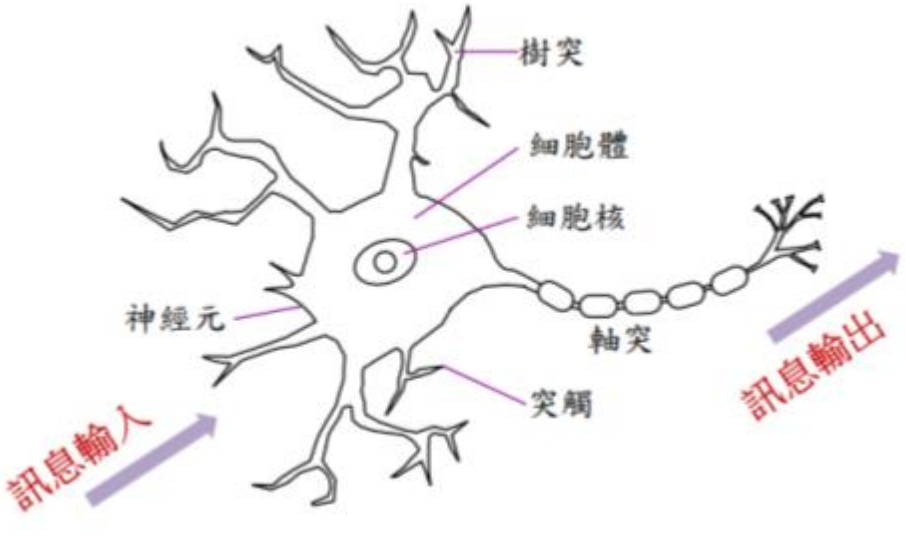
1. 類神經網路簡述

2. 類神經網路實作

本教材主要資料來源:
李宏毅 (2016), Deep Learning Tutorial. [DSC 2016] 系列活動：一天搞懂深度學習
https://www.slideshare.net/tw_dsconf/ss-62245351

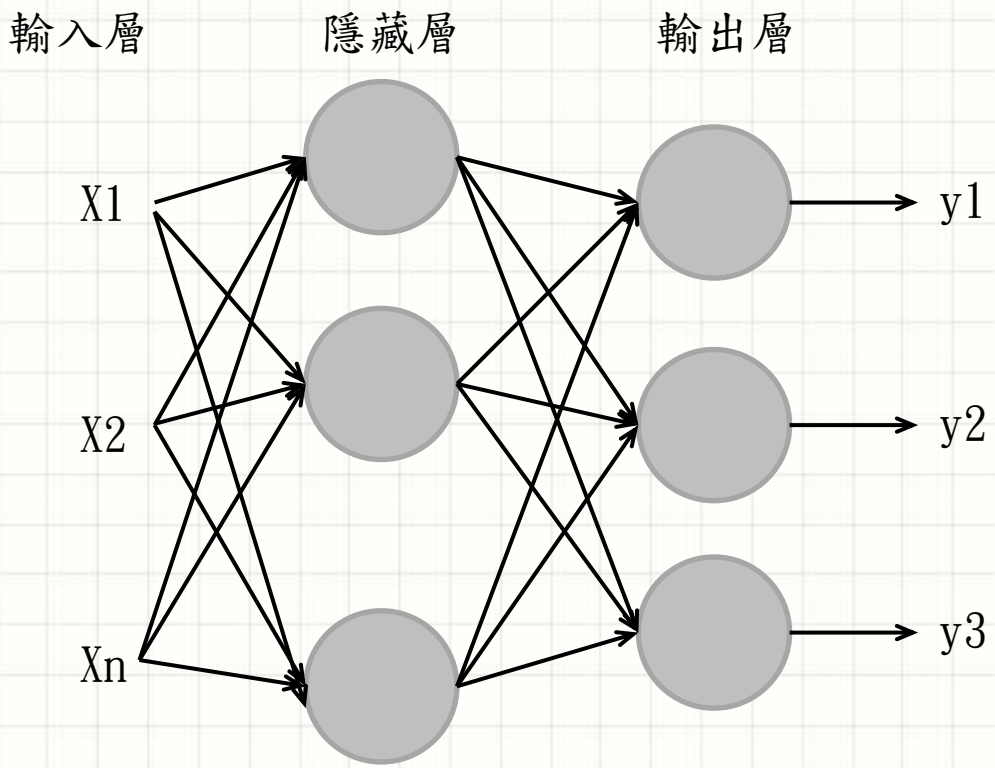
類神經網路簡述

仿造生物中神經網絡來處理數據資訊



類神經網路簡述

類神經網絡基本架構

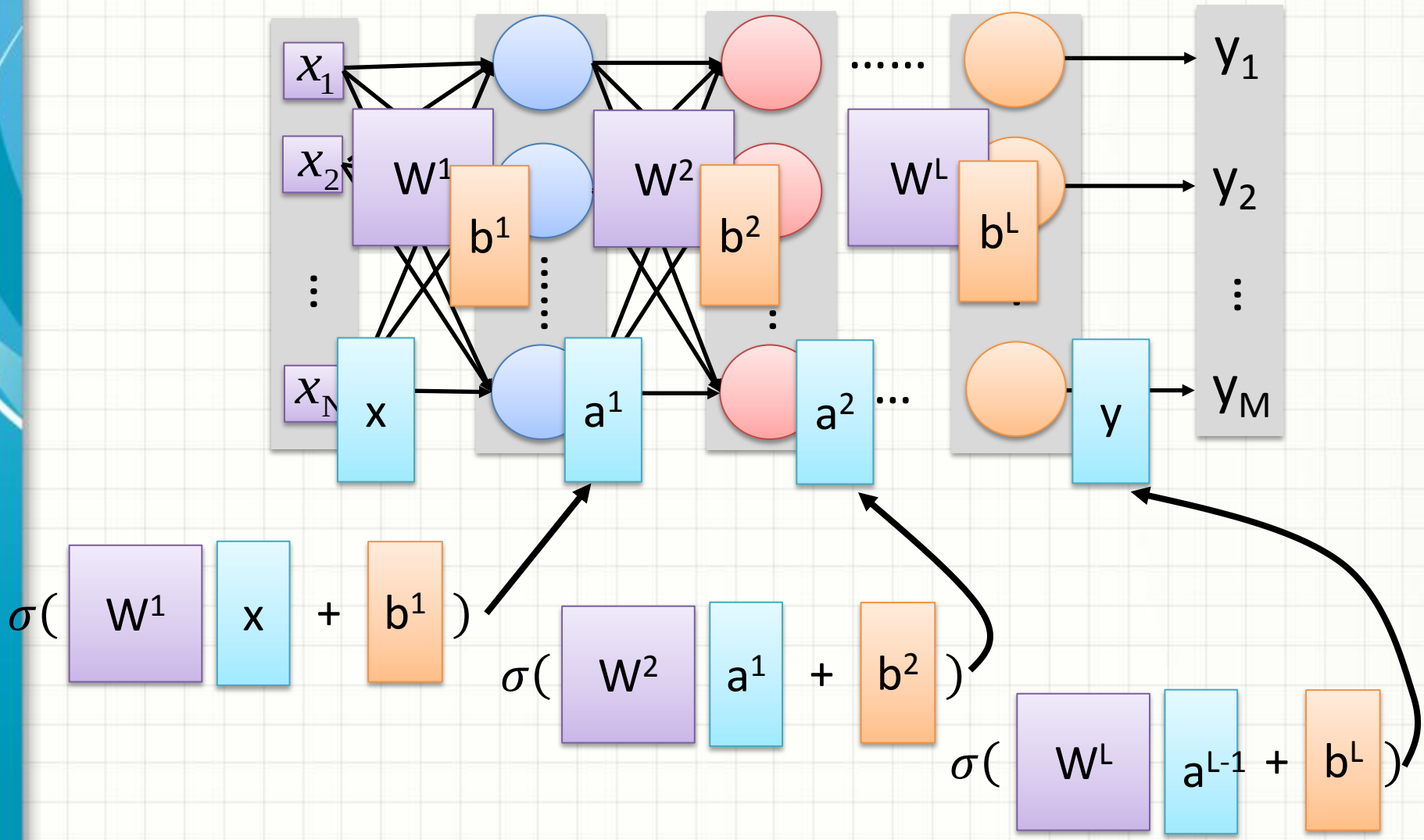


隱藏層中灰圈表示一節點

- 輸入層:接收輸入訊號。
- 隱藏層:處理神經元彼此交互作用的問題，可能有多層隱藏層，可以想成很多迴歸式在這層。
- 輸出層:處理輸出訊號。

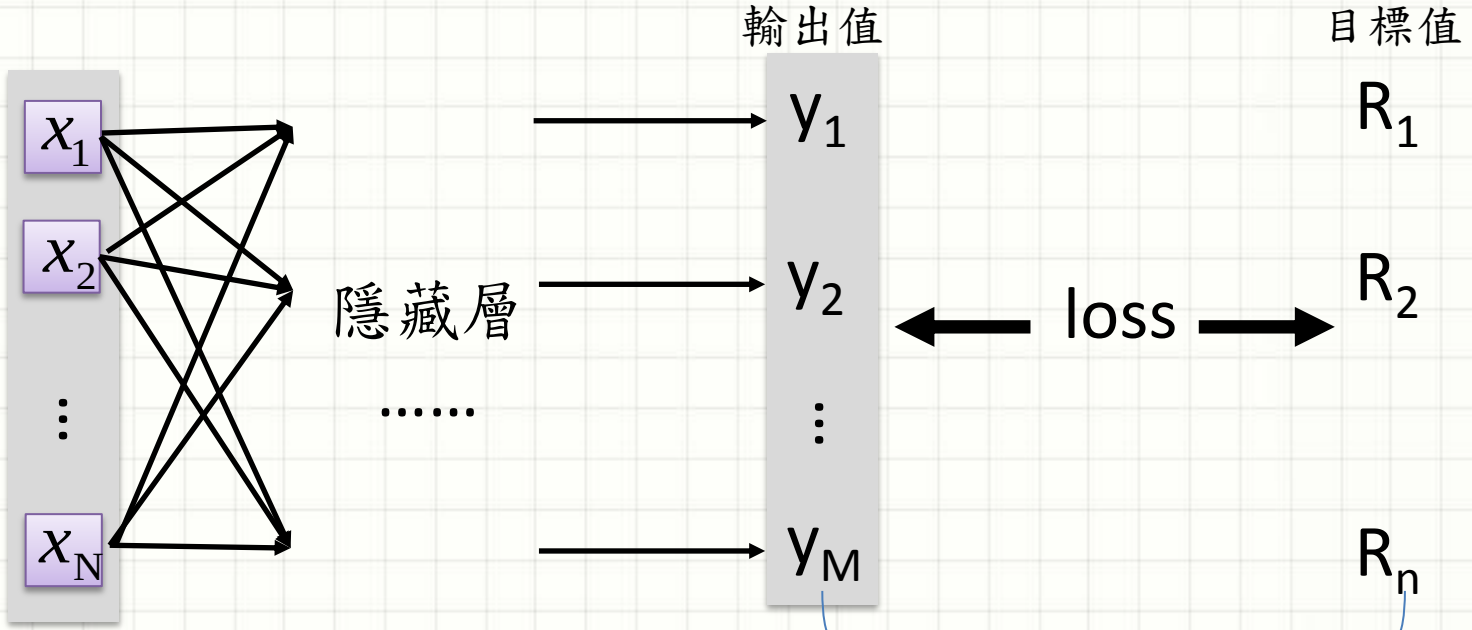
類神經網路簡述

類神經網絡基本通式



類神經網路簡述

類神經網絡基本通式



選擇適當的loss函數表達誤差，並透過更改隱藏層參數使loss最小化

$$\sum_{i=1}^{10}(y_i-\hat{y}_i)^2$$

Square Error

$$-\sum_{i=1}^{10}\hat{y}_i ln y_i$$

Cross Entropy

…等loss 函數

類神經網路簡述

類神經網絡簡易範例

weight	input	real	predict	error	new weight
(0,1,1)	(-1,2,3)	0			
	(-1,1,4)	0			
	(-1,2,7)	1			
	(-1,3,6)	1			

1. 一開始給定一初始值給weight。
2. 作用在input得到預測值。
3. 求error，也就是loss。
4. New weight 我們定義成(error * input + weight)做疊代。
5. 大於0則預測值為1，反之為0。

類神經網路簡述

類神經網絡簡易範例

第一次

weight	input	real	predict	error	new weight
(4,1,1)	(-1,2,3)	0	1	-1	(5,-1,-2)
(5,-1,-2)	(-1,1,4)	0	0	0	(5,-1,-2)
(5,-1,-2)	(-1,2,7)	1	0	1	(4,1,5)
(4,1,5)	(-1,3,6)	1	1	0	(4,1,5)

第二次

weight	input	real	predict	error	new weight
(4,1,5)	(-1,2,3)	0	1	-1	(5,-1,2)
(5,-1,2)	(-1,1,4)	0	1	-1	(6,-2,-2)
(6,-2,-2)	(-1,2,7)	1	0	1	(5,0,5)
(5,0,5)	(-1,3,6)	1	1	0	(5,0,5)

第三次

weight	input	real	predict	error	new weight
(5,0,5)	(-1,2,3)	0	1	-1	(6,-2,2)
(6,-2,2)	(-1,1,4)	0	0	0	(6,-2,2)
(6,-2,2)	(-1,2,7)	1	1	0	(6,-2,2)
(6,-2,2)	(-1,3,6)	1	0	-1	(7,-5,-4)

類神經網路簡述

類神經網絡簡易範例

第四次

weight	input	real	predict	error	new weight
(4,1,1)	(-1,2,3)	0	1	-1	(5,-1,-2)
(5,-1,-2)	(-1,1,4)	0	0	0	(5,-1,-2)
(5,-1,-2)	(-1,2,7)	1	0	1	(4,1,5)
(4,1,5)	(-1,3,6)	1	1	0	(4,1,5)

出現發散情況，需要調整參數

經過冗長try and error

類神經網路簡述

weight	input	real	predict	error	new weight
(5,1,1)	(-1,2,3)	0	0	0	(5,1,1)
(5,1,1)	(-1,1,4)	0	0	0	(5,1,1)
(5,1,1)	(-1,2,7)	1	1	0	(5,1,1)
(5,1,1)	(-1,3,6)	1	1	0	(5,1,1)

1. 所得到error最小。
2. 此範例隱藏層只有一層。
2. 此情況weight不再改變表示收斂。
3. 所得到的隱藏層算式為 $-5+1x_1+1x_2>0$

實際上這些複雜工作都是給電腦處理的

類神經網路實作

以voice數據為例https://github.com/jasonfghx/py/blob/master/python_for_class/SVM/voice.rar

Code:

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder #將類別文字轉成數字
from sklearn.preprocessing import StandardScaler #使數據標準化
#標準化在建置類神經網路上格外重要∑☆
from sklearn.model_selection import train_test_split
from sklearn import metrics
df = pd.read_csv('voice.csv')
X=df.iloc[:, :-1]
Y=df.iloc[:, -1:]
gender_encoder = LabelEncoder()
y = gender_encoder.fit_transform(Y)
scaler = StandardScaler()
scaler.fit(X)
X = scaler.transform(X)
X_train,X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,random_state=1)
#接續
```

類神經網路實作

Code:

```
from keras.models import Sequential#接下來類神經網路模型都會以keras為主軸
from keras.layers import Dense
classifier = Sequential() #建置一個初始化的類神經網路模型。
classifier.add(Dense(units = 32, kernel_initializer = 'uniform', activation
='relu',input_dim = 20))#first
#dense用於建置隱藏層，unit表示在這一層我們要給予多少node，
#首先我們建立第一層，權重是要訓練的，賦予uniform作為初始值，
#input_dim表示我們有多少變數要進入該模型。

classifier.add(Dense(units = 16, kernel_initializer = 'uniform', activation = 'relu'))
#建立第二層，16個node。

classifier.add(Dense(units = 6, kernel_initializer = 'uniform', activation = 'relu'))
#建立第三層，6個node。

classifier.add(Dense(units = 1, kernel_initializer = normal', activation = 'sigmoid'))
#建立最後一層，1個node，大部分激發函數會用sigmoid。
#接續
```

類神經網路實作

Code:

```
classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy',metrics = ['accuracy'])
#選擇優化器(rmsprop、adam…)
#loss函數(多類別選categorical_crossentropy)
print(classifier.summary())#檢視模型內容，最後確認完畢再訓練
classifier.fit(X_train, y_train, batch_size = 1, epochs = 100)#跑起來!
#batch表示進入模型訓練一批次的數據量，若為10表示一批次用10筆數據作訓練
```

```
Epoch 95/100
2534/2534 [==============================] - 1s 331us/step - loss: 0.0012 - acc: 1.0000
Epoch 96/100
2534/2534 [==============================] - 1s 330us/step - loss: 0.0099 - acc: 0.9964
Epoch 97/100
2534/2534 [==============================] - 1s 337us/step - loss: 0.0114 - acc: 0.9957
Epoch 98/100
2534/2534 [==============================] - 1s 330us/step - loss: 0.0021 - acc: 0.9992
Epoch 99/100
2534/2534 [==============================] - 1s 330us/step - loss: 0.0012 - acc: 1.0000
Epoch 100/100
2534/2534 [==============================] - 1s 331us/step - loss: 0.0011 - acc: 1.0000
Out[206]: <keras.callbacks.History at 0x153e6748>
```

```
y_pred = round(pd.DataFrame(classifier.predict(X_test)))
print(metrics.accuracy_score(y_test,y_pred))#97%準確率

#此範例為2元變數的預測
```

類神經網路實作

以fault數據為例https://github.com/jasonfghx/py/blob/master/python_for_class/tree/steeldata.rar

Code(本例為多類別預測):

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler #務必使數據標準化
from sklearn.model_selection import train_test_split
from keras.utils import np_utils, generic_utils #使y變數變成稀疏矩陣
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import  Dropout, Activation, Flatten
df = pd.read_csv('steeldata.csv')
X=df.iloc[:, :-8]
y=df.iloc[:, -1:]
scaler = StandardScaler()
scaler.fit(X)
X = scaler.transform(X)
X_train,X_test, y_train,y_test = train_test_split(X, y, test_size=0.2, random_state=1)
y_train=np_utils.to_categorical(y_train,num_classes=7)#將類別變數轉成稀疏矩陣
y_test=np_utils.to_categorical(y_test,num_classes=7)
#接續
```

類神經網路實作

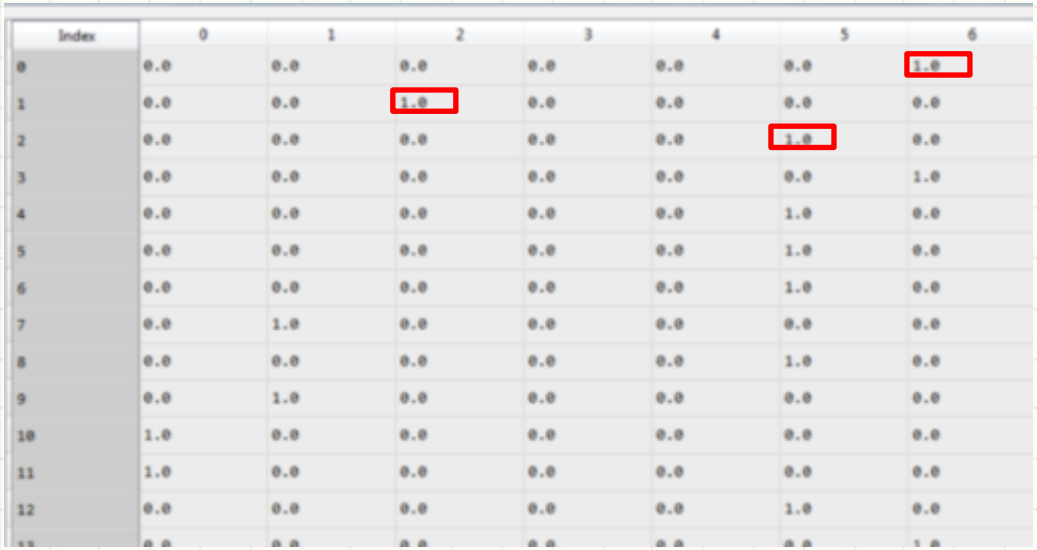
Code:

```
classifier = Sequential() #建置一個初始化的類神經網路模型。
classifier.add(Dense(units = 20, kernel_initializer = 'uniform', activation =
'relu',input_dim = 27))
#首先我們建立第一層，20個node，權重是要訓練的，賦予uniform作為初始值，
#input_dim=27表示我們有27個變數要進入該模型。
#classifier.add(Dropout(0.1))，dropout函式用來以一定機率忽略某node
#主要用於大規模的類神經網路避免overfitting
classifier.add(Dense(units = 16, kernel_initializer = 'uniform', activation = 'relu'))
#建立第二層，16個node。

classifier.add(Dense(units = 6, kernel_initializer = 'uniform', activation = 'relu'))
#建立第三層，6個node，以本數據來看，多這層導致overfitting，因此除去。

classifier.add(Dense(units = 7, kernel_initializer = 'normal', activation = 'softmax'))
#建立最後一層，因為類別只有7個要用7個node，要用softmax。
```

類神經網路實作

Code:

```
from keras.callbacks import EarlyStopping
early_stopping = EarlyStopping(monitor='acc', patience=10, verbose=2)
#patience=10表示若發現loss比上一個epoch訓練沒下降，則經過10個epoch停止訓
#練，此參數用於減少overfitting現象發生。

classifier.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics =
['accuracy'])
classifier.fit(X_train, y_train, batch_size = 50, epochs = 230,
callbacks=[early_stopping] ,validation_data=(X_test, y_test))
# 新增validation_data參數，可以一邊訓練模型，同時將測試集帶入測試準確率。
y_pred = round(pd.DataFrame(classifier.predict(X_test)))#預測
```



Index	0	1	2	3	4	5	6
0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
1	0.0	0.0	1.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	1.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	1.0
4	0.0	0.0	0.0	0.0	0.0	1.0	0.0
5	0.0	0.0	0.0	0.0	0.0	1.0	0.0
6	0.0	0.0	0.0	0.0	0.0	1.0	0.0
7	0.0	1.0	0.0	0.0	0.0	0.0	0.0
8	0.0	0.0	0.0	0.0	0.0	1.0	0.0
9	0.0	1.0	0.0	0.0	0.0	0.0	0.0
10	1.0	0.0	0.0	0.0	0.0	0.0	0.0
11	1.0	0.0	0.0	0.0	0.0	0.0	0.0
12	0.0	0.0	0.0	0.0	0.0	1.0	0.0

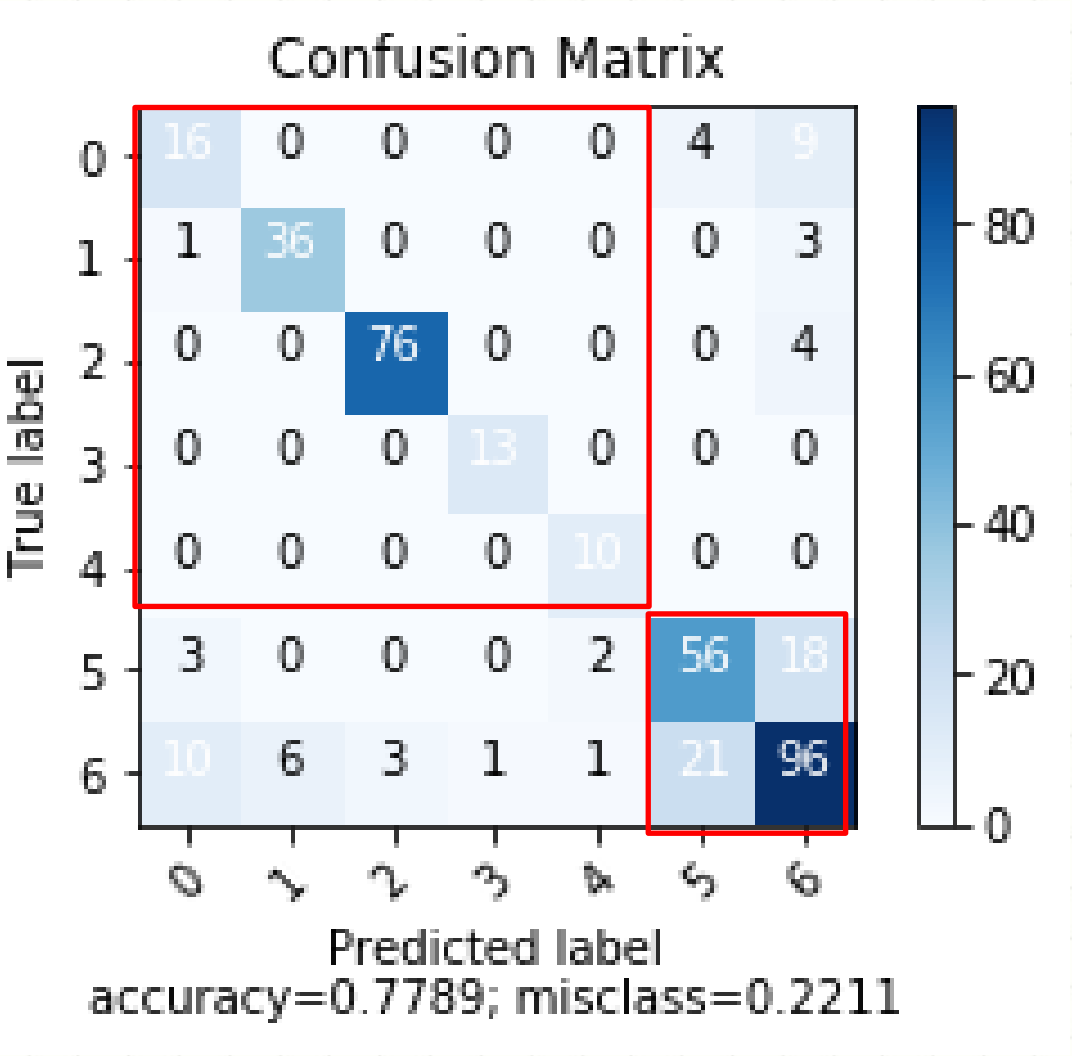
類神經網路實作

Code:

```
ab= pd.DataFrame(np.empty([389,1]))
for i in range(0,389):
ab.iloc[i,0]=pd.DataFrame(np.where(y_pred.iloc[i,:]==y_pred.iloc[i,:].max())[0]).
iloc[0,0]
print(metrics.accuracy_score(y_test,ab)) #準確率為77%

#詳細參照https://github.com/jasonfghx/py/blob/master/python_for_class/fault_NN
```

類神經網路實作

Fault數據所得到的預測與實際的混淆矩陣



類神經網路實作

1. **keras套件除了在類別變數的預測，亦能在連續變數上做回歸預測。**

2. **keras套件可以實現多個x變數預測多個y變數。**

3. **實例請參考 https://github.com/jasonfghx/python-for-data-mining/tree/master/python_for_class/NN_linearregression**