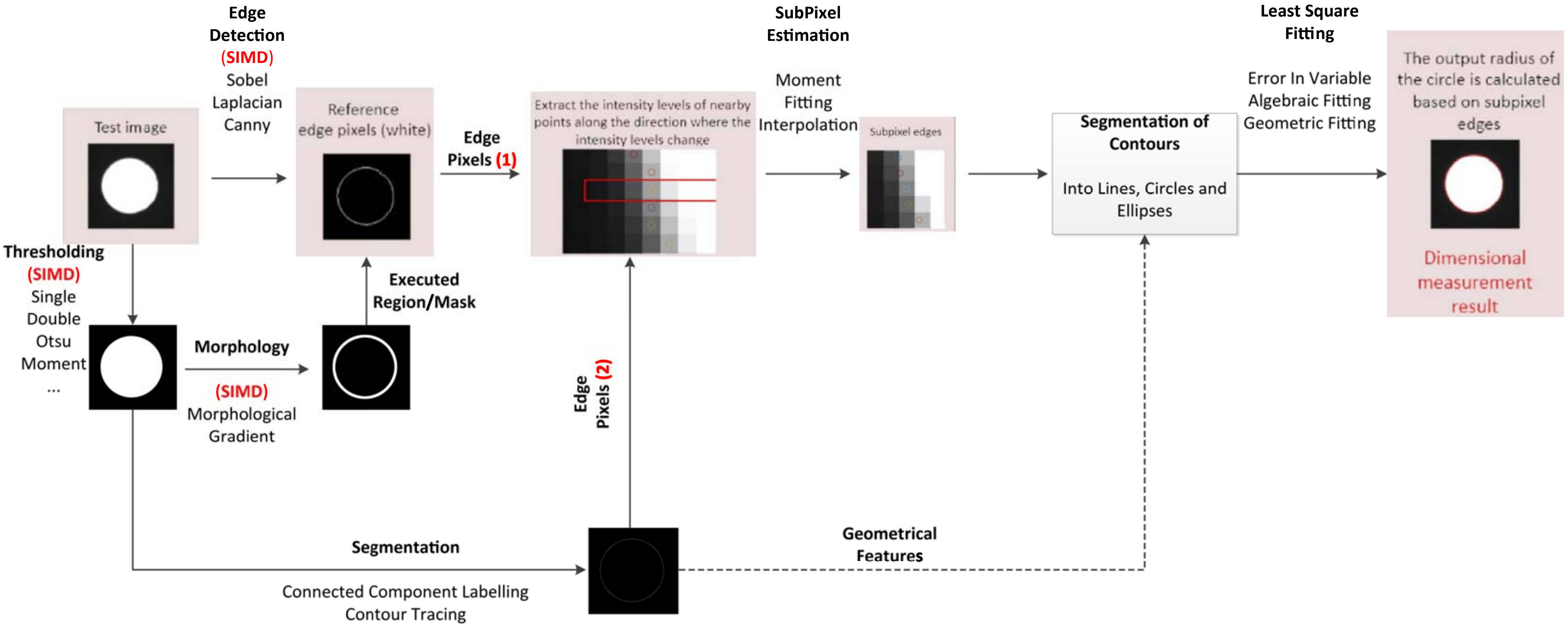
嵌入式工業機器視覺

作業二說明

大綱

- 線、圓及橢圓量測流程說明
- 處理步驟討論
  - ✓ 次像素方法
  - ✓ 直線、圓及橢圓的擬合方式
- 應用多層感知機於光學字元辨識
  - ✓ 辨識系統流程
  - ✓ 函式庫設計
- 文獻分析

線、圓及橢圓量測流程說明

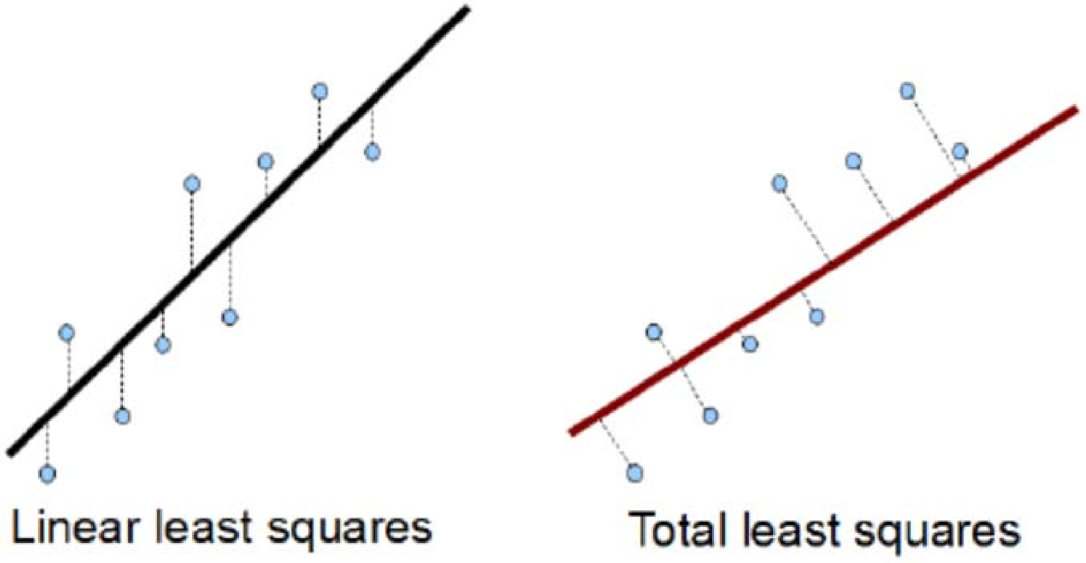


處理步驟討論

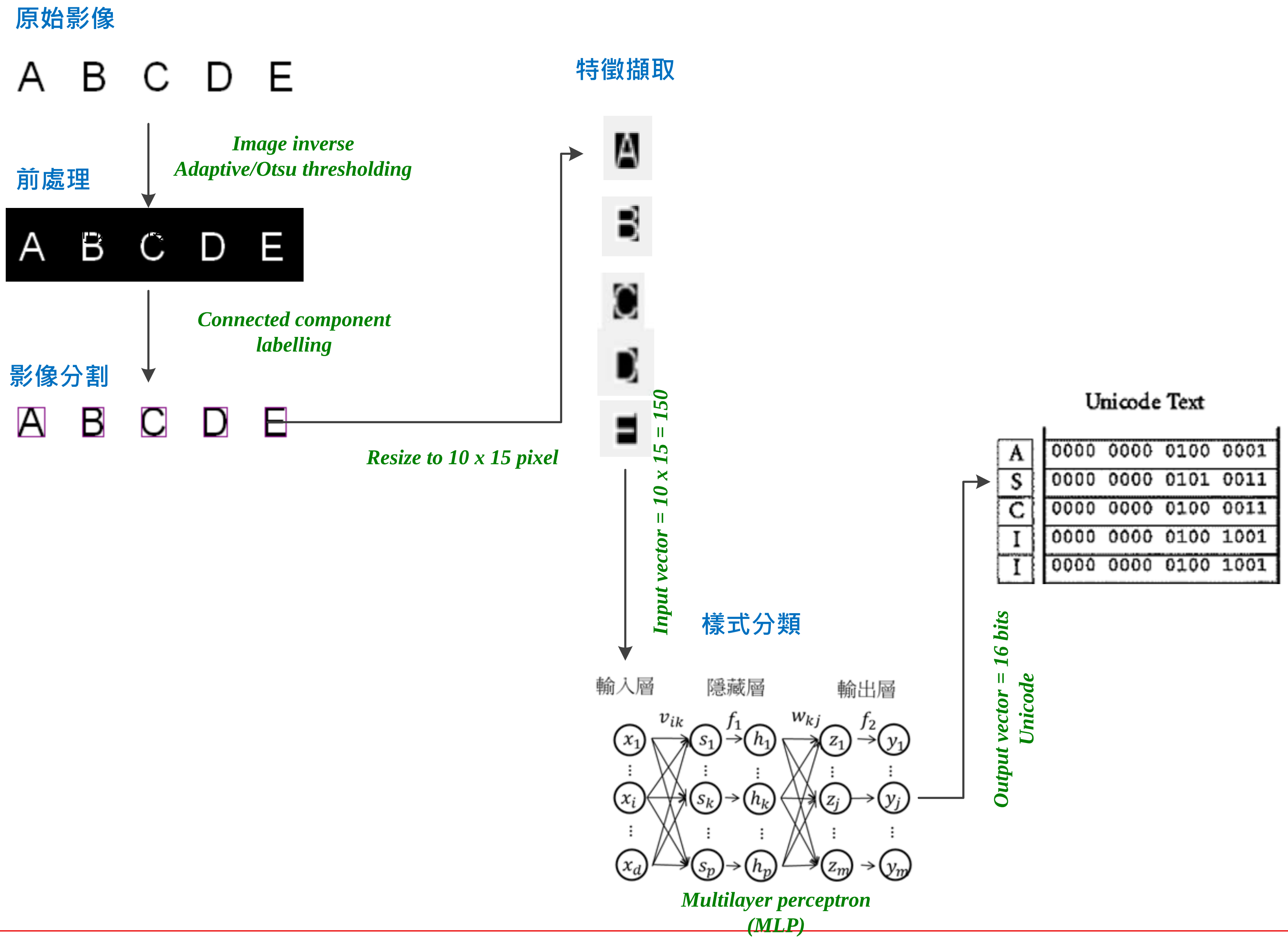
- 次像素方法：
  - Moment-based
  - Fitting-based
  - Interpolation-based / Reconstruction

處理步驟討論

- 最小平方擬合：
  - Error In Variable (EIV) / Total Least Squares (TLS)
  - Algebraic Fitting
  - Geometric Fitting



應用多層感知機於光學字元辨識



原始影像

A B C D E

前處理

Image inverse
Adaptive/Otsu thresholding

影像分割

Connected component labelling

Resize to 10 x 15 pixel

特徵擷取

Input vector = 10 x 15 = 150

樣式分類

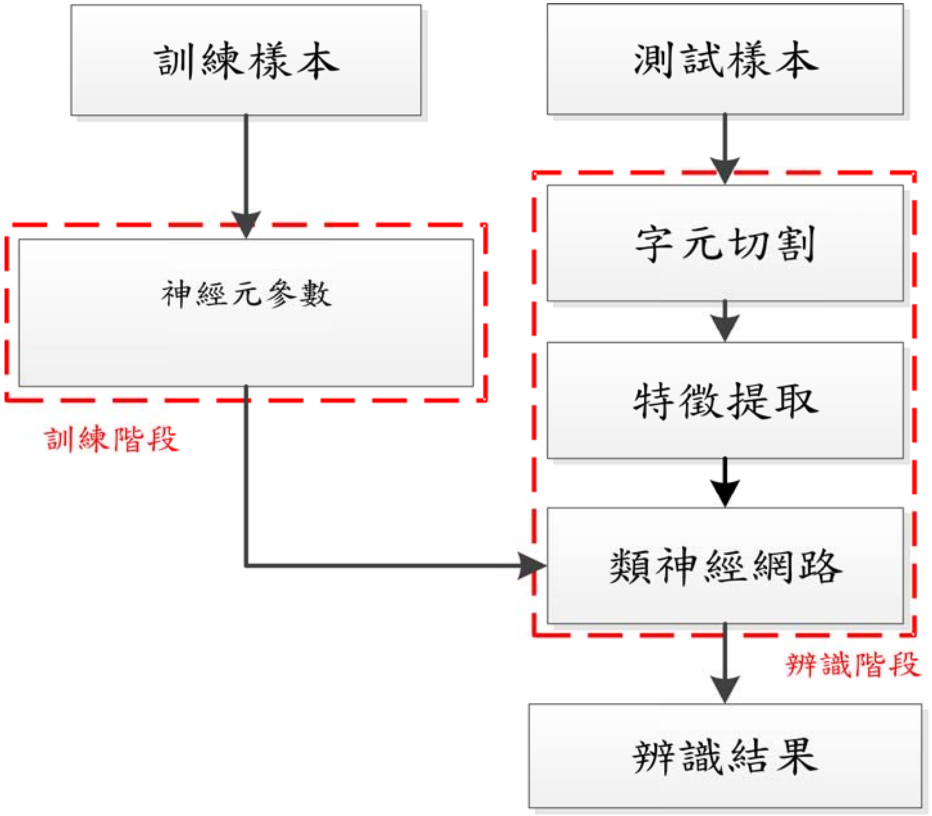
Multilayer perceptron (MLP)

Output vector = 16 bits
Unicode

Unicode Text

A	0000 0000 0100 0001
S	0000 0000 0101 0011
C	0000 0000 0100 0011
I	0000 0000 0100 1001
I	0000 0000 0100 1001

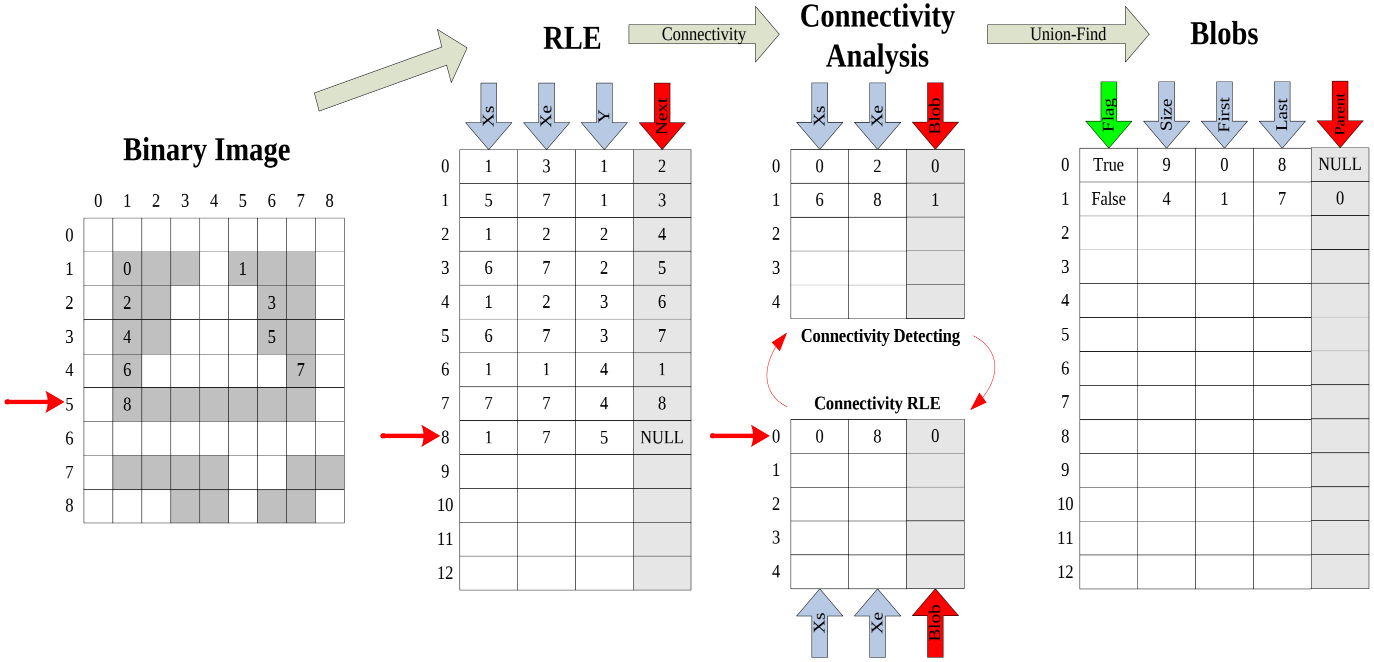
辨識系統流程



辨識系統流程(續)

➢ 物件連通法

透過建立線段編碼表的形式，使演算法可以完成物件的切割。



Binary Image

	0	1	2	3	4	5	6	7	8
0									
1		0	▨	▨		1	▨	▨	
2		2	▨				3	▨	
3		4	▨				5	▨	
4		6						7	
5		8	▨	▨	▨	▨	▨	▨	
6									
7		▨	▨	▨	▨			▨	▨
8				▨	▨		▨	▨	

RLE

	Xs	Xe	Y	Next
0	1	3	1	2
1	5	7	1	3
2	1	2	2	4
3	6	7	2	5
4	1	2	3	6
5	6	7	3	7
6	1	1	4	1
7	7	7	4	8
8	1	7	5	NULL
9				
10				
11				
12				

Connectivity

Connectivity Analysis

	Xs	Xe	Blob
0	0	2	0
1	6	8	1
2			
3			
4			

Connectivity Detecting

Connectivity RLE

	Xs	Xe	Blob
0	0	8	0
1			
2			
3			
4			

Union-Find

Blobs

	Flag	Size	First	Last	Parent
0	True	9	0	8	NULL
1	False	4	1	7	0
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					

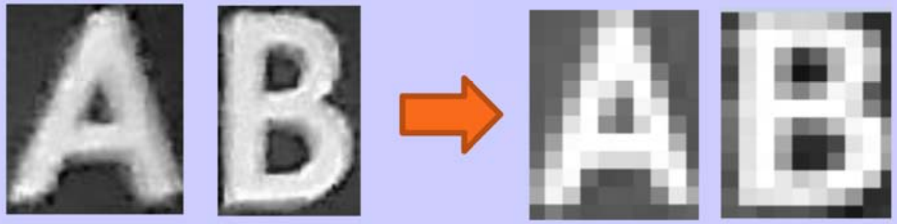
辨識系統流程(續)

- 灰階正規化

$$f(g) = a \times g + b; g \in G_{bit} \quad ; \quad a = \frac{2^{bit} - 1}{g_{\max} - g_{\min}} \quad 、 \quad b = -a \times g_{\min}$$

- 尺寸正規化

$$\begin{bmatrix} u \\ t \end{bmatrix} = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$



辨識系統流程(續)

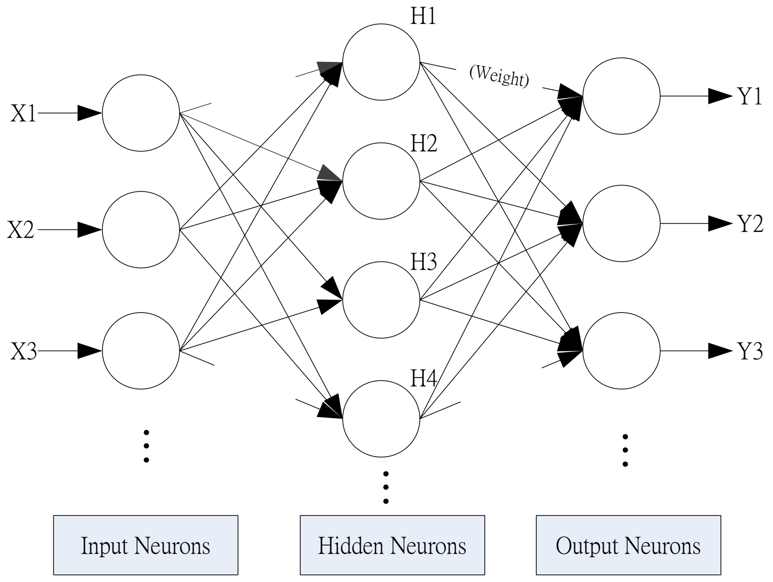
- 網路架構

  倒傳遞類神經網路為多層感知機 (MLP) 的架構

  - 前授型網路

  倒傳遞類神經網路的學習演算法為誤差倒傳遞演算法

  - 監督式學習



辨識系統流程(續)

- 網路架構

倒傳遞類神經網路為多層感知機 (MLP) 的架構

- 前授型網路

倒傳遞類神經網路的學習演算法為誤差倒傳遞演算法

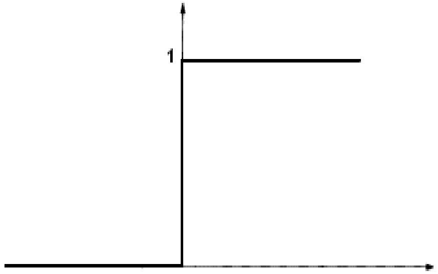
- 監督式學習

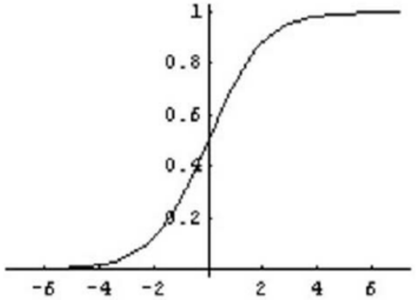
一般常見的類神經網路活化函數

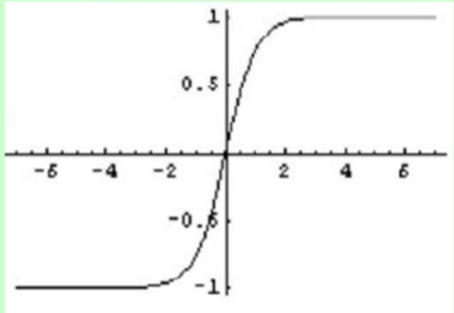
$$\begin{cases} f(\mathrm{x}) = 1, x \geq 0 \\ f(\mathrm{x}) = 0, x < 0 \end{cases}$$

$$f(\mathrm{x}) = \frac{1}{1+e^{-ax}}$$

$$f(\mathrm{x}) = \frac{1-e^{-ax}}{1+e^{-ax}}$$







辨識系統流程(續)

➢ 倒傳遞類神經網路演算法

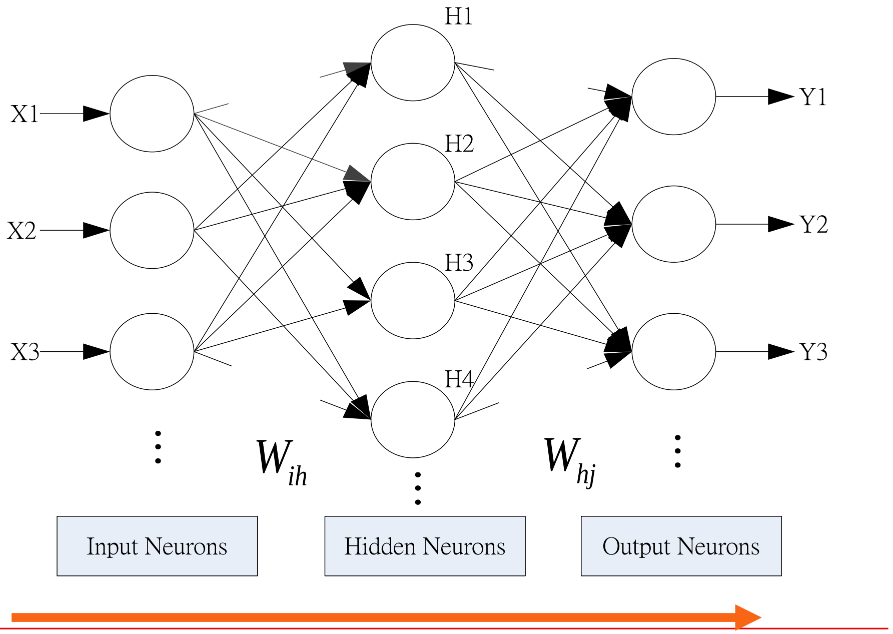
•學習過程

神經元輸入：已知 （訓練樣本）

權重值　　：未知

神經元輸出：　已知 （訓練樣本標籤）

（經由反覆的運算，學習神經元間連接的權重值）



辨識系統流程(續)

➢ 倒傳遞類神經網路演算法

•學習過程

神經元輸入：已知 （訓練樣本）

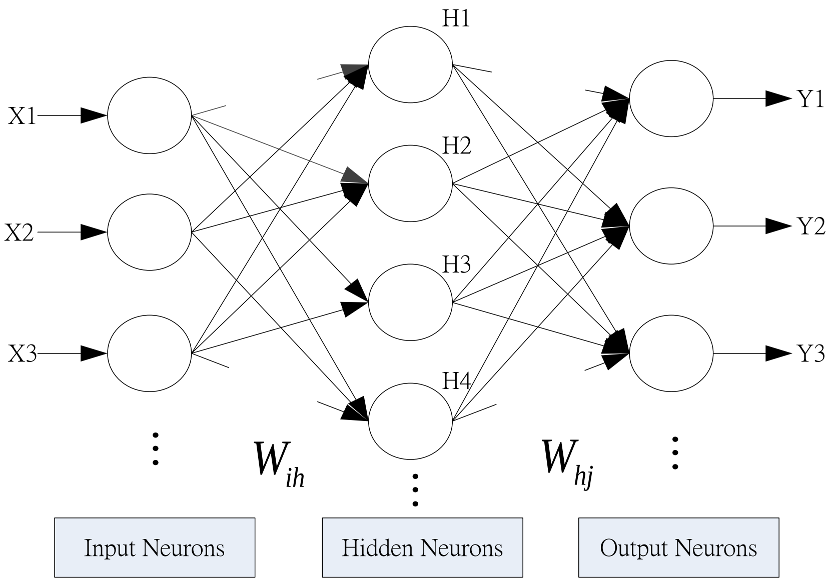
權重值　　：未知

神經元輸出：　已知 （訓練樣本標籤）

（經由反覆的運算，學習神經元間連接的權重值）

最小平方誤差： $E = \frac{1}{2}\sum_{j}^{N}(T_j - Y_j)^2$

梯度下降法： $\Delta W = -\eta \frac{\partial E}{\partial W}$



辨識系統流程(續)

➢ 倒傳遞類神經網路演算法

- 學習過程

  神經元輸入：已知 （訓練樣本）
  權重值　　：未知
  神經元輸出： 已知 （訓練樣本標籤）
  （經由反覆的運算，利用訓練樣本學習神經元間連接的權重值）

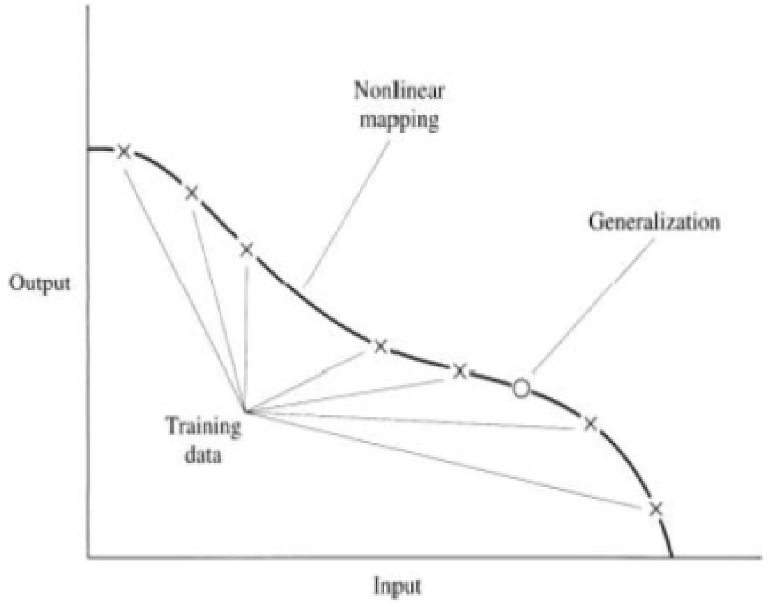
- 推論過程

  神經元輸入：已知 （測試樣本）
  權重值　　：已知
  神經元輸出： 未知
  （利用訓練出來的權重值，輸入測試樣本計算出結果）

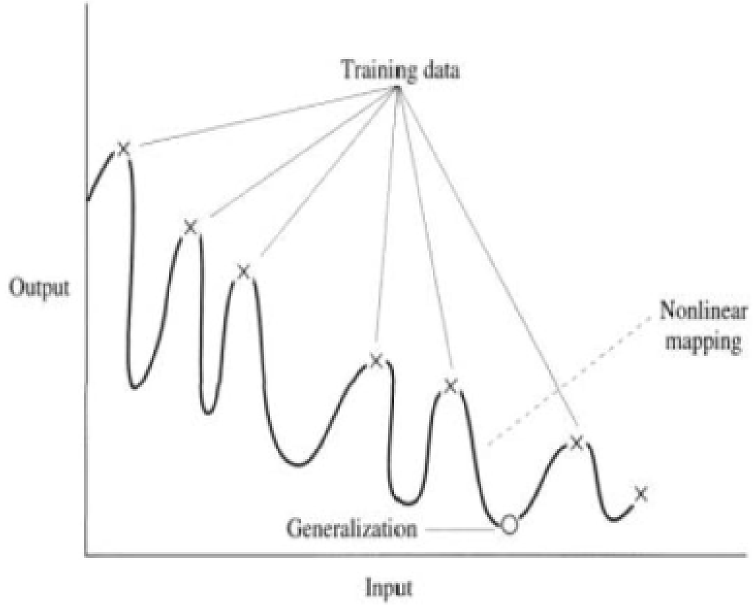
辨識系統流程(續)

- 過擬合 (overfitting)

  使用過多參數，以致於過度適應訓練資料，使模型失去它的一般化的能力。



擬合線



過擬合

辨識系統流程(續)

- 倒傳遞類神經網路隱藏層設計：
  - 方法1:

$$N_h = \sqrt{N_i \times N_o}$$

  - 方法2:

$$N_h = N_i + 1$$

  - 方法3:

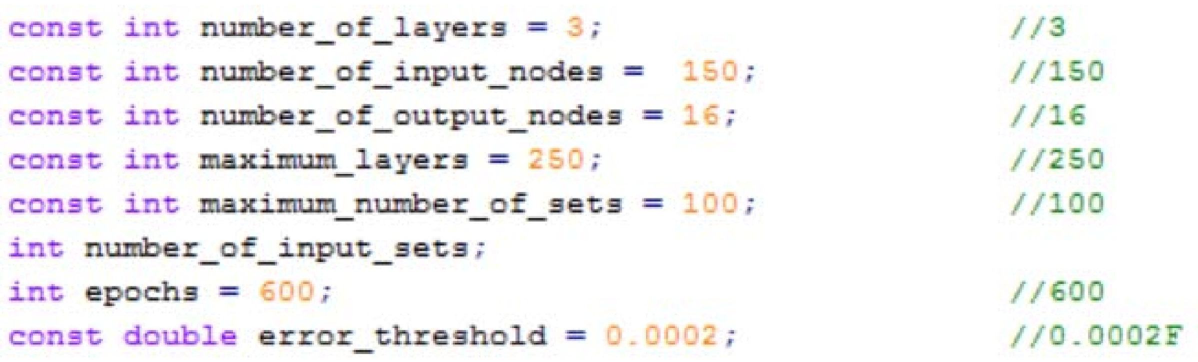
$$N_h = \frac{4n^2 + 3}{n^2 - 8}$$

1. K. Shibata and Y. Ikeda, "Effect of number of hidden neurons on learning in large-scale layered neural networks,"
2. D. Hunter, H. Yu, M. S. Pukish III, J. Kolbusz, and B. M. Wilamowski, "Selection of proper neural network sizes and architectures—A comparative study,“
3. K. G. Sheela and S. N. Deepa, "Review on methods to fix number of hidden neurons in neural networks,"

辨識系統流程(續)

✓ 倒傳遞類神經網路

- 隱藏層神經元數量
- 訓練次數
- 學習率

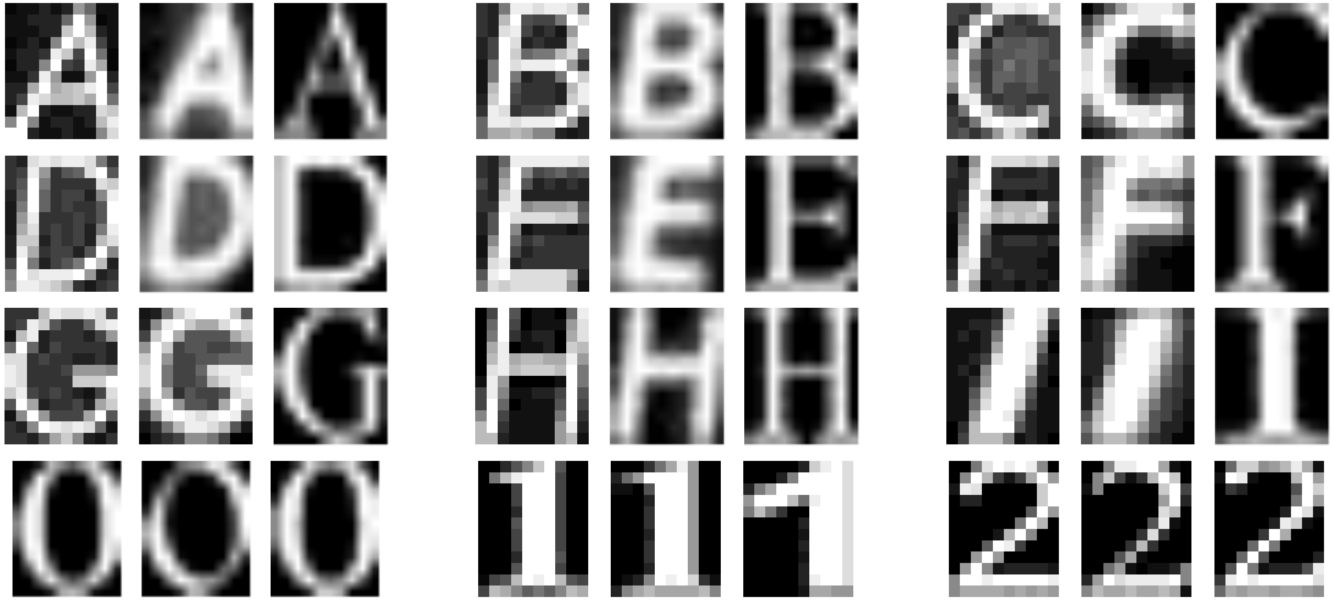


```
const int number_of_layers = 3;                     //3
const int number_of_input_nodes =  150;             //150
const int number_of_output_nodes = 16;              //16
const int maximum_layers = 250;                     //250
const int maximum_number_of_sets = 100;             //100
int number_of_input_sets;
int epochs = 600;                                   //600
const double error_threshold = 0.0002;              //0.0002F
```

辨識系統流程(續)

- 訓練樣本

  訓練樣本： xx 個 (A~Z 、 0~9 ， 36 個字元)



辨識系統流程(續)

➢ 度量標準

- 均方根誤差 (RMSE) :

$$E_{rms}=\sqrt{\frac{\sum_{p}^{M}\sum_{j}^{N}(T_j^p-Y_j^p)^2}{M\times N}}$$

- 真陽性率 (True Positive Rate ， TPR) ，亦被稱為召回率 (Recall):

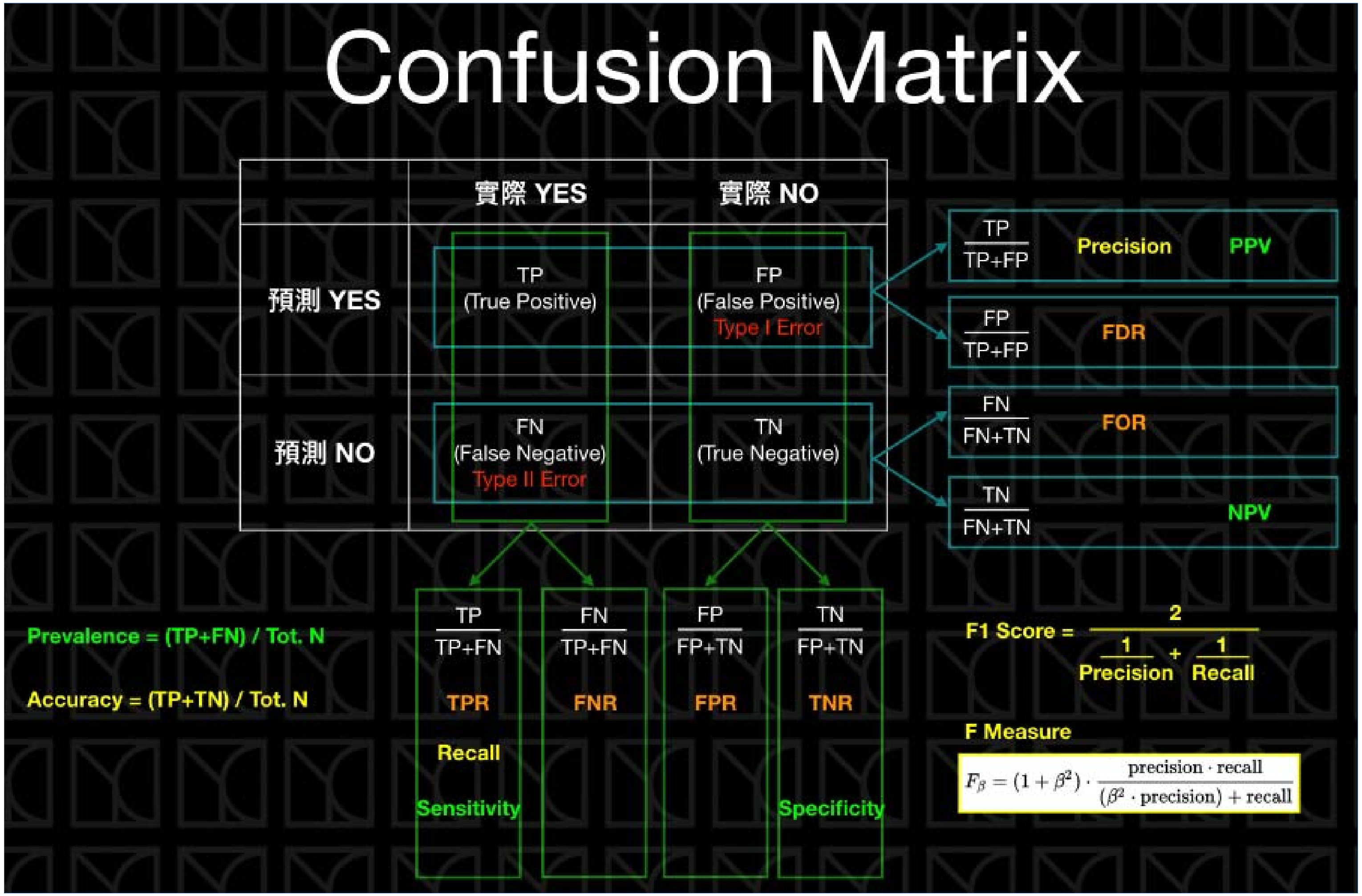
$$TPR=\frac{TP}{TP+FN}$$

- 陽性預測值 (Positive Predictive Value ， PPV) ，亦被稱為精準度 (Precision):

$$PPV=\frac{TP}{TP+FP}$$

辨識系統流程(續)

- 度量標準



Confusion Matrix

	實際 YES	實際 NO
預測 YES	TP (True Positive)	FP (False Positive) Type I Error
預測 NO	FN (False Negative) Type II Error	TN (True Negative)

$\frac{TP}{TP+FP}$	Precision	PPV
$\frac{FP}{TP+FP}$	FDR	
$\frac{FN}{FN+TN}$	FOR	
$\frac{TN}{FN+TN}$		NPV

$\frac{TP}{TP+FN}$	$\frac{FN}{TP+FN}$	$\frac{FP}{FP+TN}$	$\frac{TN}{FP+TN}$
TPR	FNR	FPR	TNR
Recall			
Sensitivity			Specificity

Prevalence = (TP+FN) / Tot. N

Accuracy = (TP+TN) / Tot. N

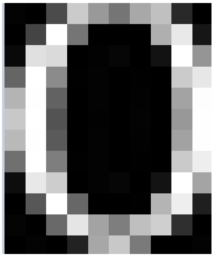
F1 Score = $\frac{2}{\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}}}$

F Measure

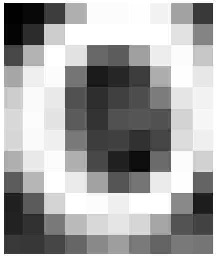
$$F_\beta = (1+\beta^2)\cdot\frac{\text{precision}\cdot\text{recall}}{(\beta^2\cdot\text{precision})+\text{recall}}$$

辨識系統流程(續)

- 相似的字元

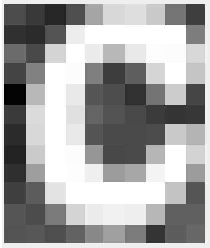


字元
0

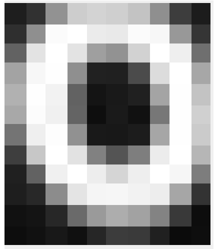


字元
O

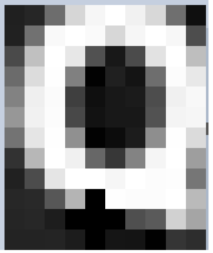
- 混淆字元



字元
C

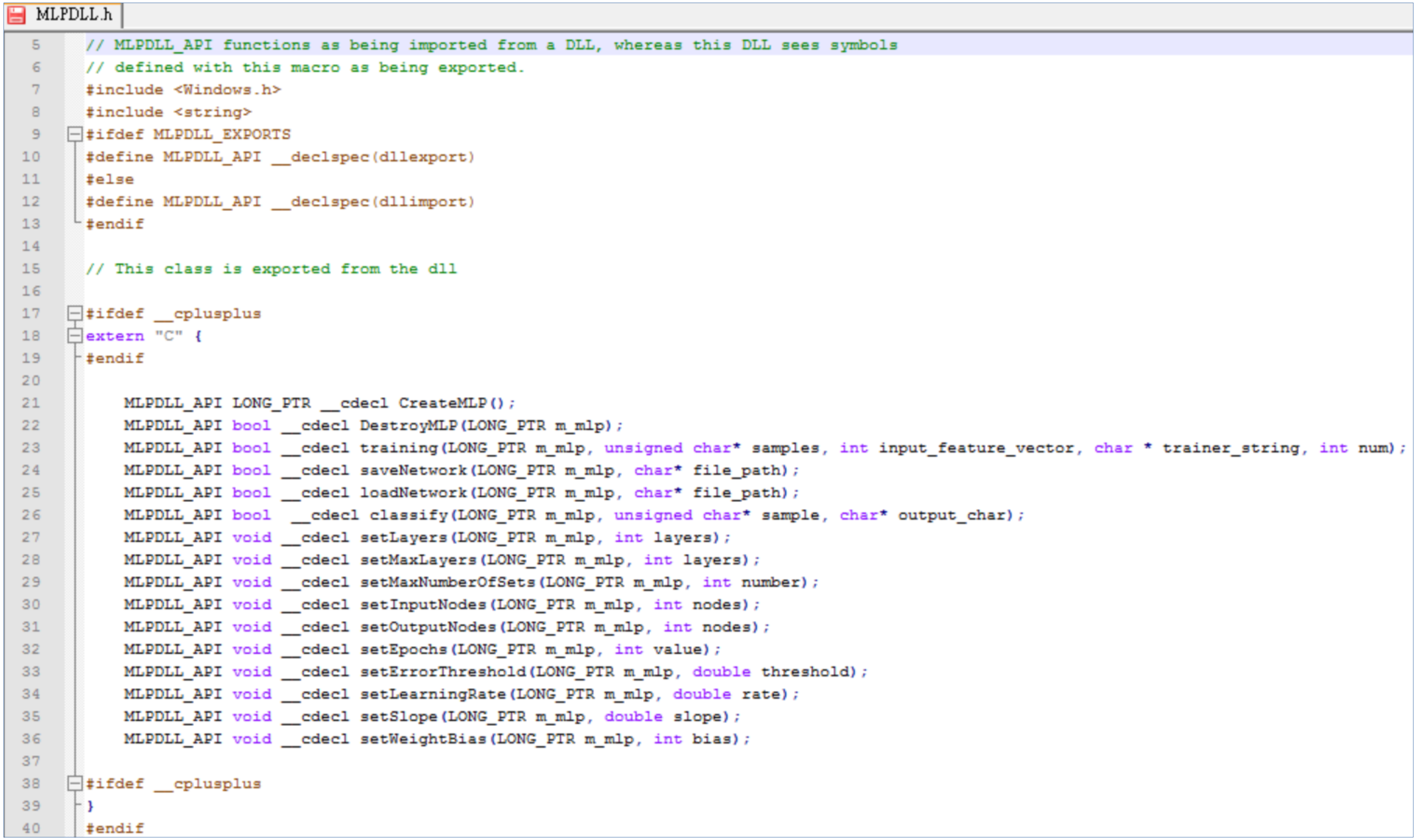


字元
O



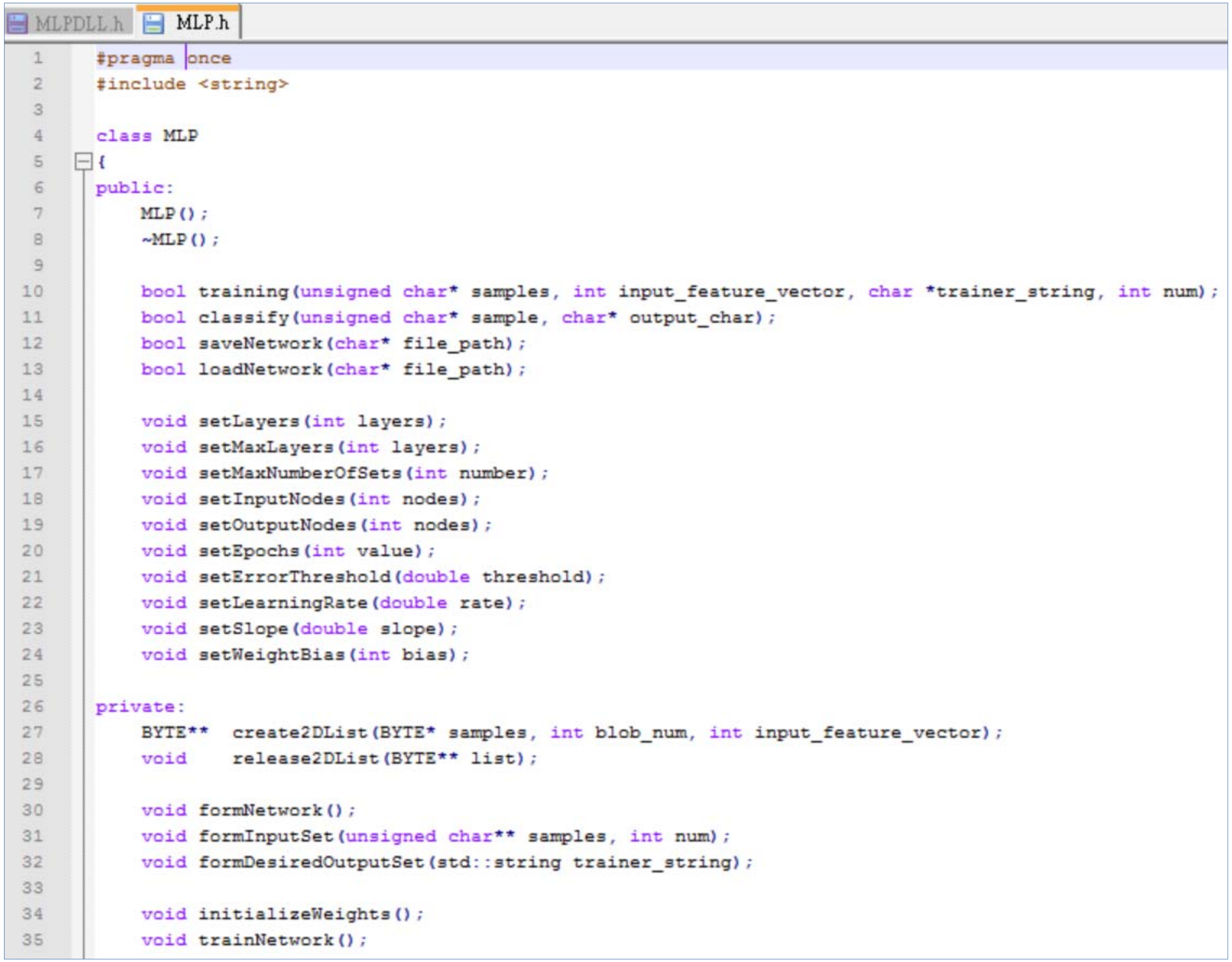
字元
Q

函式庫設計



```
// MLPDLL_API functions as being imported from a DLL, whereas this DLL sees symbols
// defined with this macro as being exported.
#include <Windows.h>
#include <string>
#ifdef MLPDLL_EXPORTS
#define MLPDLL_API __declspec(dllexport)
#else
#define MLPDLL_API __declspec(dllimport)
#endif

// This class is exported from the dll

#ifdef __cplusplus
extern "C" {
#endif

    MLPDLL_API LONG_PTR __cdecl CreateMLP();
    MLPDLL_API bool __cdecl DestroyMLP(LONG_PTR m_mlp);
    MLPDLL_API bool __cdecl training(LONG_PTR m_mlp, unsigned char* samples, int input_feature_vector, char * trainer_string, int num);
    MLPDLL_API bool __cdecl saveNetwork(LONG_PTR m_mlp, char* file_path);
    MLPDLL_API bool __cdecl loadNetwork(LONG_PTR m_mlp, char* file_path);
    MLPDLL_API bool  __cdecl classify(LONG_PTR m_mlp, unsigned char* sample, char* output_char);
    MLPDLL_API void __cdecl setLayers(LONG_PTR m_mlp, int layers);
    MLPDLL_API void __cdecl setMaxLayers(LONG_PTR m_mlp, int layers);
    MLPDLL_API void __cdecl setMaxNumberOfSets(LONG_PTR m_mlp, int number);
    MLPDLL_API void __cdecl setInputNodes(LONG_PTR m_mlp, int nodes);
    MLPDLL_API void __cdecl setOutputNodes(LONG_PTR m_mlp, int nodes);
    MLPDLL_API void __cdecl setEpochs(LONG_PTR m_mlp, int value);
    MLPDLL_API void __cdecl setErrorThreshold(LONG_PTR m_mlp, double threshold);
    MLPDLL_API void __cdecl setLearningRate(LONG_PTR m_mlp, double rate);
    MLPDLL_API void __cdecl setSlope(LONG_PTR m_mlp, double slope);
    MLPDLL_API void __cdecl setWeightBias(LONG_PTR m_mlp, int bias);

#ifdef __cplusplus
}
#endif
```

函式庫設計(續)



```cpp
#pragma once
#include <string>

class MLP
{
public:
    MLP();
    ~MLP();

    bool training(unsigned char* samples, int input_feature_vector, char *trainer_string, int num);
    bool classify(unsigned char* sample, char* output_char);
    bool saveNetwork(char* file_path);
    bool loadNetwork(char* file_path);

    void setLayers(int layers);
    void setMaxLayers(int layers);
    void setMaxNumberOfSets(int number);
    void setInputNodes(int nodes);
    void setOutputNodes(int nodes);
    void setEpochs(int value);
    void setErrorThreshold(double threshold);
    void setLearningRate(double rate);
    void setSlope(double slope);
    void setWeightBias(int bias);

private:
    BYTE**  create2DList(BYTE* samples, int blob_num, int input_feature_vector);
    void    release2DList(BYTE** list);

    void formNetwork();
    void formInputSet(unsigned char** samples, int num);
    void formDesiredOutputSet(std::string trainer_string);

    void initializeWeights();
    void trainNetwork();
```

文獻分析

- 次像素方法(列舉數篇，可另外尋找適合的文獻)：
  - **Subpixel Edge Localization Based on Adaptive Weighting of Gradients.**
  - **Non-linear fourth-order image interpolation for subpixel edge detection and localization.**
  - **Sub-pixel edge detection based on an improved moment.**
  - **Sub-pixel edge contour detection algorithm based on Cubic B-Spline interpolation.**
  - **Accurate subpixel edge location based on partial area effect.**
  - **The accuracy of sub-pixel localization in the Canny edge detector.**

文獻分析

- 最小平方擬合法(列舉數篇，可另外尋找適合的文獻)：
  - Least squares fitting of circles and lines.
  - Estimation of planar curves, surfaces, and nonplanar space curves defined by implicit equations with applications to edge and range image segmentation.
  - Circular and linear regression: Fitting circles and lines by least squares.
  - The information of algebraic fitting and geometric fitting can refer to https://people.cas.uab.edu/~mosya/cl/CPPcircle.html. That introduces the algorithms of Kasa fit, Pratt fit, Taubin fit and Hyper fit.

文獻分析與研究方法

- □ 為何要做文獻分析，以及如何學習研究方法，請參考：