=====================================================================

0. Introduction

=====================================================================

Feature Extraction & Feature Selection

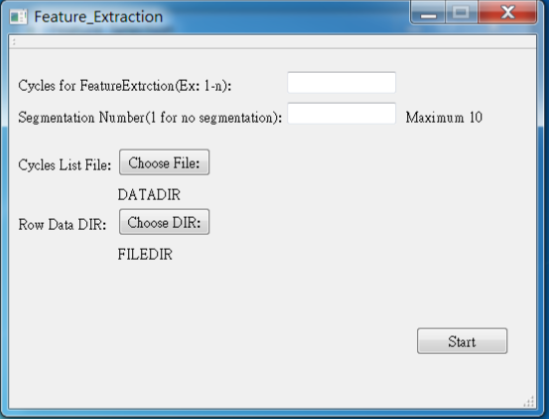
=====================================================================

1. Program Usage

=====================================================================

GUI 版本：

Feature\_Extraction：



Cycles for FeatureExtraction (Ex: 1-n)：欲使用的Cycle數目，如第一個cycle輸入1-1，1~5 cycles輸入1-5，以此類推。

Segmentation Number(1 for no segmentation)：在每個run裡面進行切分動作再計算feature，1 表示不切分，最大值為10。

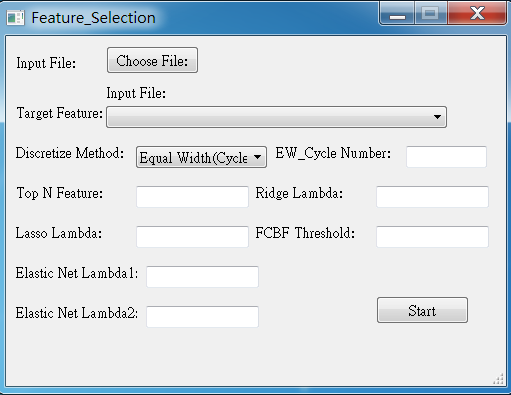
Cycle List File：點選Choose File按鈕，選取use\_file\_list.csv檔案所在位置，選取後路徑將顯示於下方。

Row Data DIR：點選Choose File按鈕，選取ROW Data所在資料夾，選取後路徑將顯示於下方。

點選Start後將執行，成功出現”Finish”字樣的對話框，失敗則為”Fail”。

Output 為一至兩個csv file，如為no segmentation output 1 file，如有segmentation output兩個file，分別為切分後兩種擺法(增加資料量(直擺)，增加資料維度(橫擺))。

Feature\_Selection：



Input File：點擊Choose File:按鈕，選取Feature\_Extraction所輸出的檔案，檔案名稱將於選取後顯示於下方。

Target Feature：自動讀取檔案內Feature名稱，可挑選其一作為分析的目標值。(於此數據中應是挑選dP\_Filter相關數值)

Discretize Method：

1. Equal Width(Cycle) 等距切割進行數值離散化，於後方輸入離散化數值數目。
2. Manual 手動指定離散化切割點。(EX：5,10,15)

Top N Features：輸入n，輸出將會輸出前N項選出來的Feature。(註：某些演算法選出來的Feature可能只有少數個，不一定有N個)

Ridge Lambda：進行Ridge迴歸的Lambda值，1-3。

Lasso Lambda：進行Lasso迴歸的Lambda值，1-3。

Elastic Net Lambda 1：進行Elastic Net迴歸的Lambda1值，1-3。

Elastic Net Lambda 2：進行Elastic Net迴歸的Lambda2值，1-3。

點選Start後將執行，成功出現”Finish”字樣的對話框，失敗則為”Fail”。

最後產生下列四個csv檔案：

FSo\_Details: 紀錄使用的特徵、平均距離、標準化參數、回歸係數

FSo\_DiscretizedData: 離散化的資料

FSo\_NormalizedData: 標準化的資料(含回歸合成函數)

FSo\_Result: 篩選排名結果

BASH版本：

FE\_no\_GUI.exe input\_file dir(rowdata) Cyclebegin(num) Cycleend(num) segmentnum

1. input\_file1 use\_file\_list.csv絕對相對路徑皆可

2. input\_file1 rowdatarowdata 資料夾位置絕對相對路徑皆可

3. Cyclebegin 起始cycle number

4 .Cycleend 結束cycle number

5. segmentnum 等份數

共5個參數

FS\_no\_gui.exe Output\_seg2\_1-23\_2.csv "dP\_Filter\_max" "ew\_cycle=4" 0.1 1 1 1 1 10 false 15 "FSi\_exclude\_list.txt" "FSi\_use\_feature\_list.txt"

|  |  |  |
| --- | --- | --- |
| 共13個參數 | | |
| 1. | input\_file | FE處理後的檔案(Output\_noSeg.csv, Output\_seg1.csv, Output\_seg2.csv) |
| 2. | target\_feature | 目標特徵("dP\_Filter (X1)\_max") |
| 3. | disct\_method | 指定離散化方式("ew\_cycle=4", "5,15,20") |
| 4. | fcbf\_thrd | MI-FCBF演算法之threshold (0.01) |
| 5. | ridge\_lambda | Regression-RIDGE演算法之lambda(1, 2, 3) |
| 6. | lasso\_lambda | Regression-LASSO演算法之lambda(1, 2, 3) |
| 7. | els\_lambda1 | Regression-ElasticNet演算法之lambda1(1, 2, 3) |
| 8. | els\_lambda2 | Regression-ElasticNet演算法之lambda2(1, 2, 3) |
| 9. | print\_n | 列印前n名的結果(1~n) |
| 10. | discretize by cycle | 是否對cycle進行discretize，true表是，false表對全部的run進行discreteization |
| 11. | top\_k | 演算法篩選的個數(15,20,30 ...) |
| 12. | exclude\_file | 分析時排除之變數(txt檔，一行一個參數) |
| 13. | include\_file | 分析時納入之變數(txt檔，一行一個參數，ALL表示使用全部參數) |

=====================================================================

2. Build from Source Code

=====================================================================

FeatureExtraction:

1.gsl 1.15 64bit https://code.google.com/p/oscats/downloads/list

說明：解壓放C根目錄 Compile 時 linker 下指令連結此函式庫

EX: ../../../../GSL-1.15/lib/libgsl.a ../../../../GSL-1.15/lib/libgslcblas.a

2.QT 設計gui

32bit 5.3 mingw 4.8.2

default dll linking

如果要standalone的exe,需built static qt

附上 built 好的static版 如無法使用需重built 參考以下

http://qt-project.org/wiki/How-to-build-a-static-Qt-for-Windows-MinGW

3.DataBase:

如需使用快速讀取功能(csv.h),compile時加以下指令

-DUSE\_FAST\_CSV

FeatureSelection:

Regression based:

mlpack(使用mingw make)

1.libxml libconv：

下載位置(直接使用prebuilt的只能使用.dll.a版本)

<ftp://ftp.zlatkovic.com/pub/libxml/64bit/>

或是自己重編static版(我是自己編)

<http://stackoverflow.com/questions/3429101/building-the-latest-iconv-and-libxml2-binaries-in-win32>

linker指令(如果需要的話)

..\libxml2-2.9.1-win32-x86\lib\libxml2.dll.a ..\libxml2-2.9.1-win32-x86\lib\libxml2.a

..\libiconv-1.8-20020830\lib\libiconv.a ..\libiconv-1.8-20020830\lib\libcharset.a

2.armadillo-4.320.0(enable lapack/blas)

如何enable lapack/blas：

include\armadillo\_bits\config.hpp

#define ARMA\_USE\_LAPACK

#define ARMA\_USE\_BLAS

這兩行uncomment

linker指令

..\Lapack\_win32\_release\blas\_win32\_MT.lib ..\Lapack\_win32\_release\lapack\_win32\_MT.lib

並把兩dll放到執行檔旁

3.何處下載lapack/blas

<http://ylzhao.blogspot.tw/2013/10/blas-lapack-precompiled-binaries-for.html>

4.boost

可以自己編或使用含boost的mingw

含boost mingw :

<http://nuwen.net/mingw.html>

自己編:

<http://stackoverflow.com/questions/20265879/how-to-build-boost-1-55-with-mingw>

<http://www.boost.org/>

4.mlpack本身(很難編...)

參考

<http://www.mlpack.org/trac/wiki/MLPACKOnWindows>

所需套件下載完後(lapack 64bit,libxml2 64bit)

注意Cmakelists裡面comment掉下面這兩段段再用cmake:

#if (WIN32)

# link\_directories(${Boost\_LIBRARY\_DIRS})

# set(Boost\_LIBRARIES "")

#endif (WIN32)

#add\_definitions(-DBOOST\_TEST\_DYN\_LINK)

cmake裡可加上下列兩個entry

Boost\_NO\_BOOST\_CMAKE

Boost\_USE\_STATIC\_LIBS

另外，如果compile之後錯在tree\_test.cpp

comment 掉log::的部分(共有兩行)

MI based:

FEAST:

選用部分header、cpp檔運用於Feature\_Selection內。

Src資料夾：

ArrayOperations.cpp

CalculateProbability.cpp

Entropy.cpp

MutualInformation.cpp

Include資料夾：

ArrayOperations.h

CalculateProbability.h

Entropy.h

MutualInformation.h

FSAlgorithms.h

FSToolbox.h

MIToolbox.h