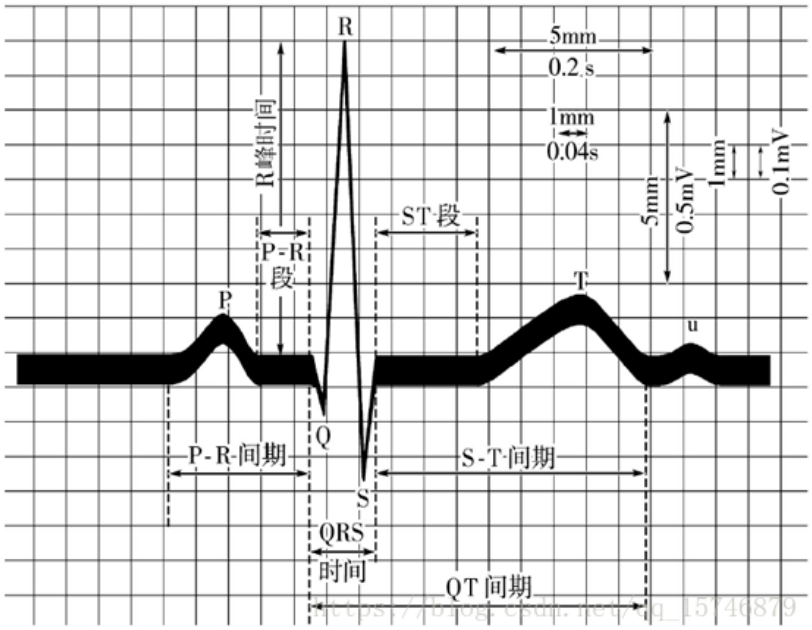
ECG ×AI: 機器/深度學習的 ECG 應用入門 (1)

前言

你好，我是研究 ECG 算法的搬磚工 Winham。目前搞這個方向已經挺長時間了，總想著把自己的一些入門經驗分享一下，卻不知道從何下手。說實話，關於 ECG 算法的研究相對冷門一些，網絡上系統的資料也比較少，有的多是故作高深的論文。想想當時入門時真是走了不少彎路，也真心體會到如果能有一個相對系統一點的教程或是博客，再配合一些可以直接上手的代碼，對於初學者來說是很重要的。所以寫了這幾篇渣文，內容簡短，簡單，都是這個方向的基礎。不求大家看了覺得多好，只求可以讓初學的童鞋少走一點彎路。涉及到的代碼也都開源了，放在了我的 github 上：https://github.com/Aiwiscal/ECG-ML-DL-Algorithm-Matlab 。代碼為 Matlab 寫成。由於涉及到了機器/深度學習，建議還是有這方面的知識比較好。說起機器/深度學習，Python 更適合一點，後續 Python 版本的代碼也會放出來。不過就本人體驗來說，入門的話，還是 Matlab 更適合一些。

ECG，是 electrocardiogram 的縮寫。就是我們平時常見的心電圖。典型的心電信號由 P 波，QRS 波，T 波等一系列特徵波組成，它們以及一些特徵段（QRS 間期，ST 段，PR 段等）包含著豐富的病理信息：



醫生就是通過分析這些波形的特點，結合自己的經驗給出診斷結果。近幾年，隨著人工智能的興起，“智能醫療”的概念也火了起來。說白了，就是用一些機器學習或深度學習算法，學習數據中的規律，能夠像醫生一樣給出一個準確的診斷結果。想像一下，其實這件事情如果真做成了，那自然是一件大好事，畢竟醫療資源是很緊缺的。不過目前挑戰還是很大的……有點扯遠了，還是說回ECG。我們的目的，就是研究如何有效利用機器/深度學習算法以及一些數字信號處理算法，使用 ECG 信號，實現對一些心血管異常的診斷。
接下來的幾部分將會從數據庫的獲取，QRS 波定位，傳統機器學習，深度學習四個方面對算法的設計和應用進行探討，所涉及內容均為基礎，並有配套源代碼，並且推薦了一部分代表性文獻供研究。有些地方可能會有錯誤和疏漏，請諒解。另外，算法的設計，有時候很依賴於個人的經驗，其具體原理很難闡述地非常清楚，需要長時間積累才能領會。希望大家好運!

ECG ×AI：機器/深度學習的 ECG 應用入門

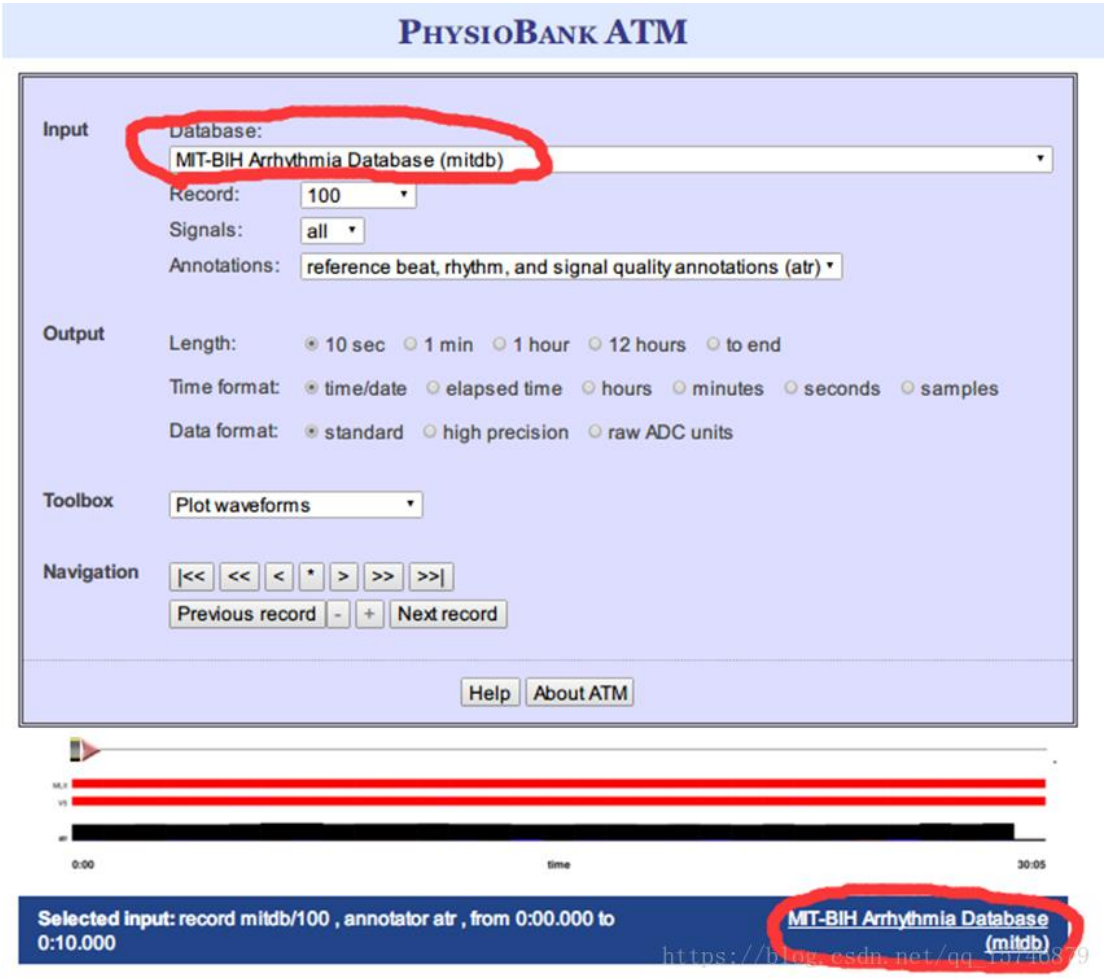
(2)

ECG 數據庫：解決數據來源問題

1.數據庫簡介與獲取

經過上一章節的介紹，相信你對於我們的研究對象——心電信號 ECG 已經有了一個初步的認識，也清楚了我們的目標就是使用機器/深度學習算法，能自動識別指定的 ECG 類型，從而實現“人工智能”式的診斷（汗。。。）。“巧婦難為無米之炊”，我們的“米”（ECG 數據）從哪裡來呢？當然有喜聞樂見的數據庫供我們使用。

首先貼一個網址：https://www.physionet.org/cgi-bin/atm/ATM



這個網站上匯集了許許多多生理信號，除了我們熟悉的 ECG，還有腦電信號 EEG，心音信號 PCG 等等。“弱水三千，只取一瓢”，這裡我們主要來說一下 ECG 領域最常用（沒有之一）的數據庫：MIT-BIH Arrhythmia Database。如上圖所示，點擊“Input”選擇框，下拉選擇 MIT-BIH Arrhythmia Database（注意，有很多 MIT-BIH 打頭的數據庫，不要搞錯了），“Input”選擇框下面還有幾個選擇框，可以分別選擇記錄，信號，標記等等，選定後頁面上會有一個區域展示一小段信號。如果想詳細了解並下載數據，點擊上圖右下角紅圈中的鏈接可以跳轉至數據庫專屬頁面。

詳細的介紹可以自己去看，全英文，英語不好的建議配合有道詞典。在這裡只說一些重要的信息：MIT-BIH Arrhythmia Database（以下簡稱 MITdb）包含了 48 條雙導聯 ECG 記錄，其中，除少數記錄外，每條記錄的第一個導聯都是 II 導聯，每條記錄長度為 30 分鐘，採樣率 360 Hz，算起來有 30*60*360=648000 個採樣點，但其實每條記錄都有 650000 個點，也就是並非嚴格等於 30 分鐘。不過這沒什麼很大影響。打開 MITdb 的專屬頁面後，很容易就發現了大量數據文件的鏈接，48 條記錄（不連續編號 100-234），每一條記錄對應著 3 個文件，就像這樣：

各個文件的作用，已經展示的很清楚。.atr：標記文件，保存著人工標註的心拍位置和類型。.dat:數據文件，保存著我們需要的心電信號。.hea：頭文件，保存著這條記錄的附加信息。這三個文件中頭文件（.hea）可以直接打開，其餘兩個不可直接打開，強行記事本打開後看到的都是亂碼。打開頭文件後，一般看到的是這樣的：

```
100 2 360 650000
100.dat 212 200 11 1024 995 -22131 0 MLII
100.dat 212 200 11 1024 1011 20052 0 V5
# 69 M 1085 1629 x1
# Aldomet, Inderal
```

第一行含義為：記錄編號導聯數採樣率採樣點數。第二、三行則說明了.dat 文件的信息，它是 212 格式編碼的等等。其詳細的說明請參照文獻：（宋喜國, 鄧親愷. MIT-BIH 心率失常數據庫的識讀及應用[J]. 中國醫學物理學雜誌, 2004, 21(4):230-232.）當然，如果你對這些東西並不關心也可以不看，對我們後面的內容影響不大。至於這個數據庫中到底包含了哪些我們感興趣的類型，作為我們分類識別的目標，這個我們用到的時候再說。

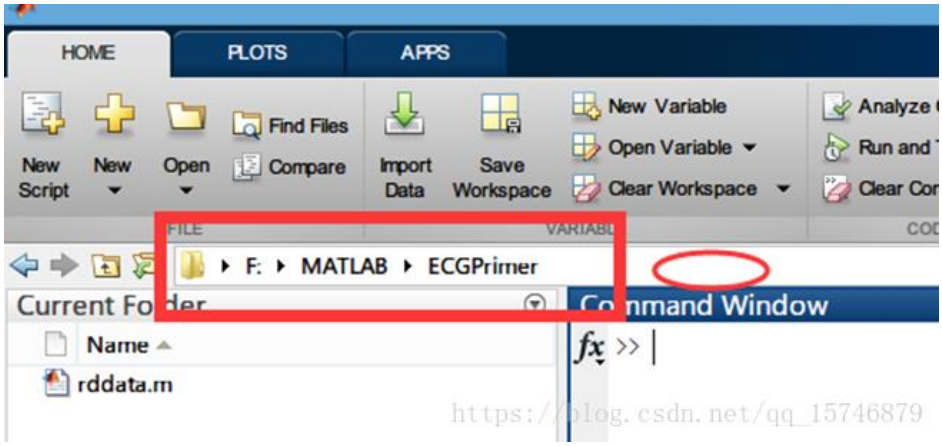
說了半天，似乎我們一直在紙上談兵，目前手裡什麼也沒有。一個不好的消息：如果你不會一些下載技巧，要下載這 48*3=144 個文件，你需要每個都點一下，因為官方似乎並沒有提供一個完整的包。這一點其實不用擔心，本人已經下載好並且打好了包提供給各位下載。方式是大家喜聞樂見的百度網盤：鏈接：https://pan.baidu.com/s/1BG1sPsWO8Ey2GdAepUa1GQ 密碼：z3ey

2.數據庫的 Matlab 讀取

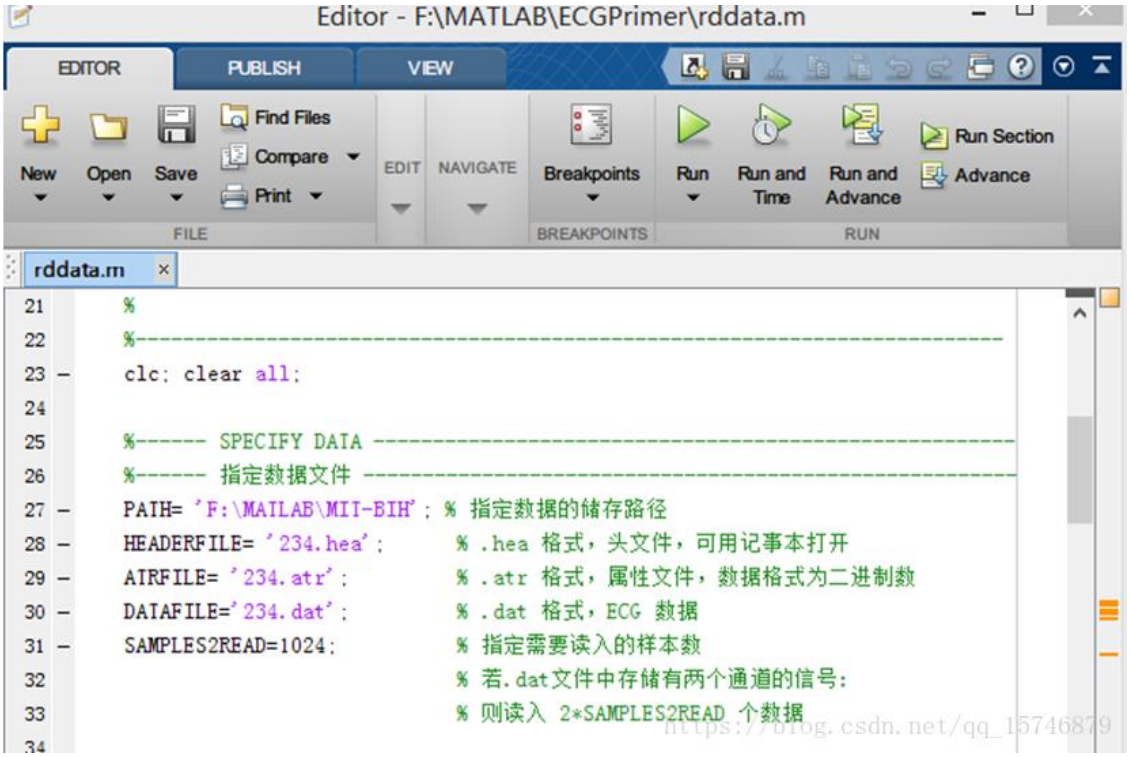
如果你閱讀了上一段建議的，對於 MITdb 數據格式說明的論文，十有八九會對讀取數據庫感到困難。這種格式並不常見，而且對於初學者並不友好。自己編程實現數據庫讀取的門檻又比較高，不是每個人都能實現，而且我們的目標也不是數據的編解碼研究。還好有大佬已經為我們寫好了讀取數據的代碼，我們只需要 run 一下就基本達成目的了。目前該代碼已經放到了我的 github 上(文後有地址），名為 rddata.m：

簡單說下怎麼用，建議單獨建一個文件夾作為我們後續代碼的存放處，例如這裡我建立了一個名為 ECGPrimer 的文件夾，路徑為：F:\MATLAB\ECGPrimer。把 rddata.m 文件放入該文件夾。

1）打開 matlab（這里為 2013a 版本，其餘版本類似），點擊界面偏上方顯示路徑的框的偏右空白處（例如下圖中橢圓處），輸入我們剛才的路徑，回車。當前路徑就成為了我們剛才建立的文件夾，左邊可以看到我們之前放入的 rddata.m 文件，雙擊打開。

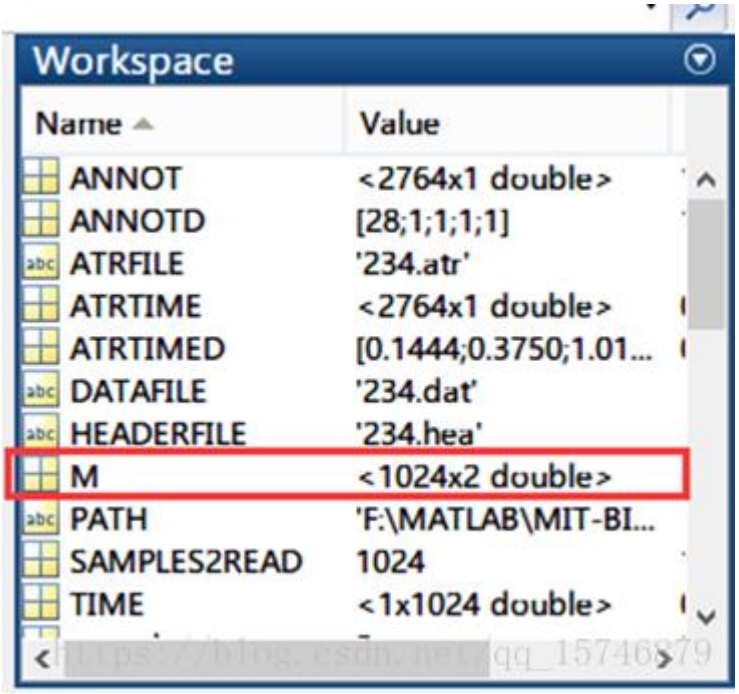


2） 打開 rddata.m 後，注意一開始的關於數據庫文件的存放路徑，以及指定文件名的設置，如圖：

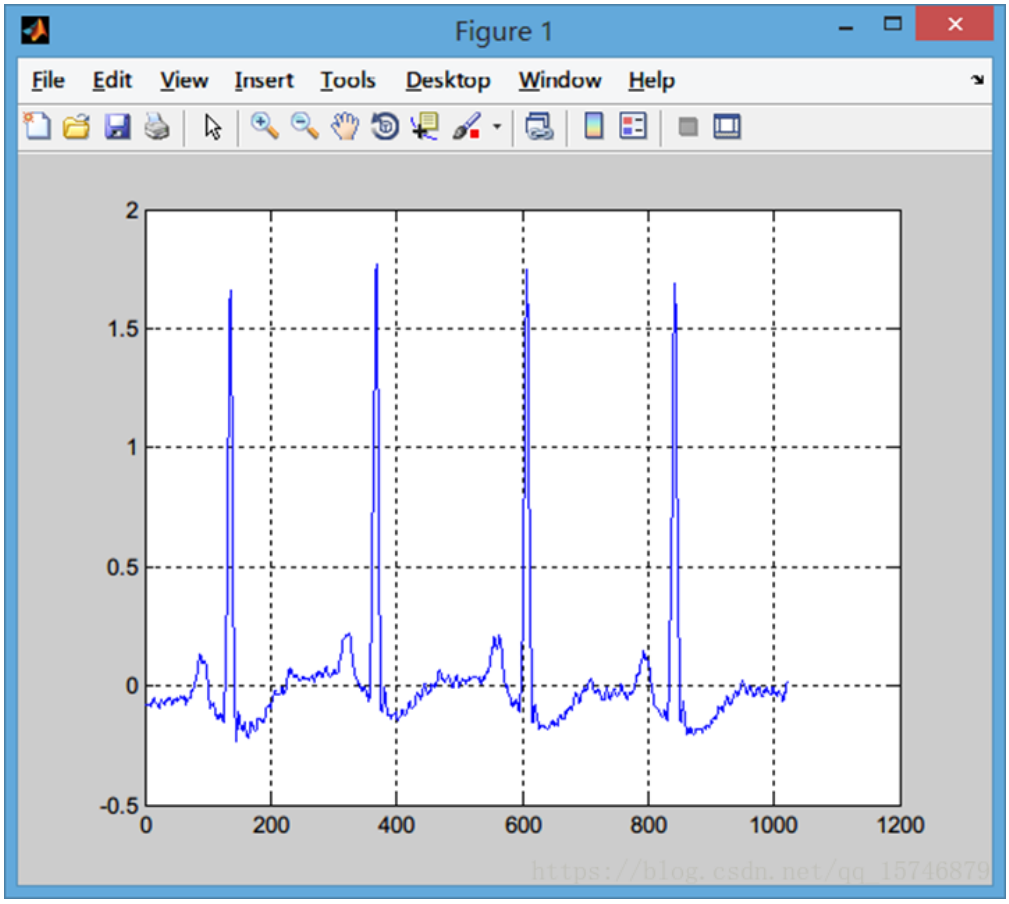


例如圖中我把下載好的 144 個數據庫文件解壓，放在了 F:\MATLAB\MIT-BIH，然後需要讀取編號為 234 的記錄，指定讀取的點數為 1024。點擊控制欄的 run 按鈕，相關數據會被讀入到 matlab 的工作空間 workspace 中。與此同時，matlab 會生成一張信號圖，上面同時繪製了兩個導聯的信號，以及一些標記的數字，這裡我們先關掉不理會。

3）然後我們觀察 matlab 的 workspace，此時應該如下圖所示：



注意，此時變量名為 M 的矩陣中包含了我們需要的 ECG 信號，尺寸為 1024（指定讀取點數）*2（導聯數）。我們只關心第一個導聯的信號，在 matlab 命令窗口裡輸入：plot(M(:,1));grid on;可繪製出第一個導聯的信號：



至此，我們已經成功將數據庫中的 ECG 信號讀取至 matlab 中。另外，workspace 中其餘 2 個矩陣也值得我們注意。如 3）中的 workspace 截圖，一個是 ANNOT，存儲了該記錄中各心拍的人工標註類型代碼，例如有：

% 注释代码		说明	
% 0		No TQRS	
% 1	N	Normal beat	正常搏动
% 2	L	Left bundle branch block beat	左束支传导阻滞
% 3	R	Right bundle branch block beat	右束支传导阻滞
% 4	a	Aberrated atrial premature beat	异常房性早搏
% 5	V	Premature ventricular contraction	室性早搏

完整的對比說明表也已經上傳至本人 github：Data Instructions.txt。ANNOT 的意義在於提供給了我們各心拍專家診斷的結果，用於後續深度學習算法的標籤。另外一個是 ATRTIME，它存儲了各心拍的人工標註位置，一般以 QRS 波為基準。即該記錄的幾分

幾秒出現了一個心拍。而這個矩陣的意義在於後續幫助我們評估心拍定位算法的準確性，或是直接根據它提供的人工標註位置截取心拍。

3.小結

本節我們學習了 ECG 數據庫的獲取與 matlab 讀取，解決了數據來源問題。如果你對其他數據庫也感興趣，可以在上面介紹的網站中繼續了解。然而現在還不宜直接進行機器/深度學習模型的開發，還有那些重要的事情沒做？且聽下回分解。

ECG ×AI：機器/深度學習的 ECG 應用入門（3）

QRS 心拍定位：解決識別對象問題

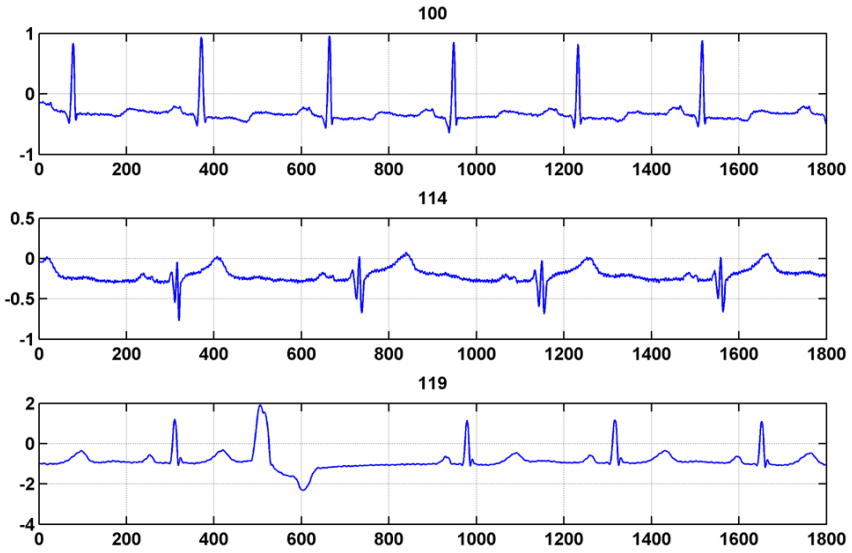
1.引言

上一節我們解決了數據來源問題,成功將 ECG 數據讀入了 Matlab 中.而我們現在還不能馬上使用高大上的機器/深度學習算法,原因在於一個問題:在這個問題中,對於我們的機器/深度學習來說,直接處理的對像是什麼?
是我們在上一節說的一條"記錄"嗎?對於大多數研究/論文來說,並不是所謂的記錄,而是 ECG 的基本單位:心拍,就是在最開始的時候介紹的,包含了 P 波，QRS 波，T 波的信號段。那為什麼不是“記錄”呢？我猜測的原因可能如下：
1）一條充分長的記錄中包含的信息太多，可能包含了很多種類型的心拍，直接全局處理不利於精細的識別。
2）一條充分長的記錄數據量偏大，若一次性全局投入後端機器/深度學習算法，不利於復雜而又大量的迭代。

3）目前可用的開源數據庫數據量偏少，若直接以一條記錄為一個算法處理單位，則總體樣本量大大下降，不利用機器學習，尤其是深度學習的應用。以 MIT-BIH 數據庫為例，若以記錄為單位處理，則只有 48 個樣本；若以心拍為單位處理，樣本量可達 10W+。

暫時看不懂也沒關係，總之記住，我們在這裡直接處理的是心拍，而非一整條記錄。

那麼問題就來了，我們讀到 Matlab 中的 ECG 數據都是以記錄形式存在的，包含了形態各異，數量不等的心拍，例如下圖給出了 100，114，119 信號的一部分，我們怎麼樣才能定位提取這些心拍呢？



這就引出了 ECG 算法領域的經典問題：QRS 波定位檢測問題。通過檢測 QRS 波的位置，可以大體定位心拍所在的位置。

2.基本思想

1）為什麼是 QRS 波？

直觀來看，各類型心拍的各波段中，QRS 波往往是最大、最明顯的、最尖銳的，本身就利於檢測；而在數學上，稱為 QRS 波的“奇異性”，表現為斜率的突變，存在不可導的點，即波峰（或波谷）點。利用該性質可配合多種處理手法，如差分法，小波變換法等等進行有效檢測。

2）基本步驟

預處理+自適應閾值。預處理的目的在於兩點：

(1) 消除噪聲和其他的雜波（P 波，T 波）等。

(2) 同時使波形模式變得更為單一，QRS 波的變化更為突出。

預處理完成後，配合自適應閾值完成 QRS 波的定位。自適應的意思是能夠適應不同類型的記錄。自適應機制越好的算法，能夠在各種類型的記錄下良

好地區分目標與非目標，從而達到很高的準確率和可用度。自適應閾值的設計原則，個人總結如下：

(1) 自適應閾值變換機制中的經驗參數需固定。在自適應閾值的變化機制中，有些參數可能需要我們根據經驗提前設定好。這沒有問題。但是一旦確定好之後檢測性能時，對於不同的記錄，這些經驗參數不可以再重新設定。

(2) 自適應閾值要跟隨波形實時地，穩健地變化。自適應閾值，並不是至給一整條記錄計算好一個閾值後就一成不變了，而是要跟隨波形實時地變化，例如當檢測到的 QRS 波振幅呈總體上升態勢時，自適應閾值也要相應地提高，反之則要相應降低；自適應閾值的變化往往不可過於劇烈，可通過設定上下限的方式使其變化更加穩健。

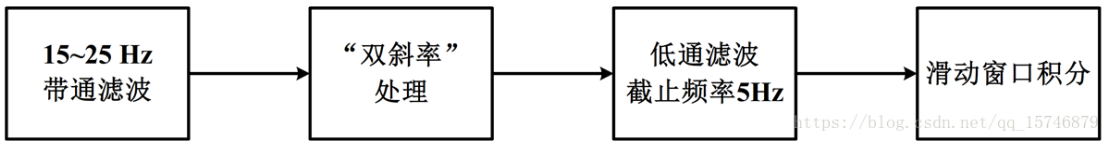
(3) 自適應閾值要有針對性地防錯檢、漏檢的機制。例如可以回溯之前的波形，再次進行檢測防止漏檢；利用“不應期”（當一個 QRS 波出現後，短時間（~0.24s）內不會再出現第二個）的含義，在檢測到一個 QRS 波後短時間內不再檢出 QRS 波，防止錯檢。當然並不是說有了這些機制就完全可以避免錯檢漏檢，而是有效地減少。

以上幾條關鍵點是本人自己總結的，可能有所疏漏，僅供參考。另外這些所謂目的和原則對於初學者來說都過於抽象，而且往往在閾值的設計中交織在一起。下面一部分我們將結合具體算法來進行講解。

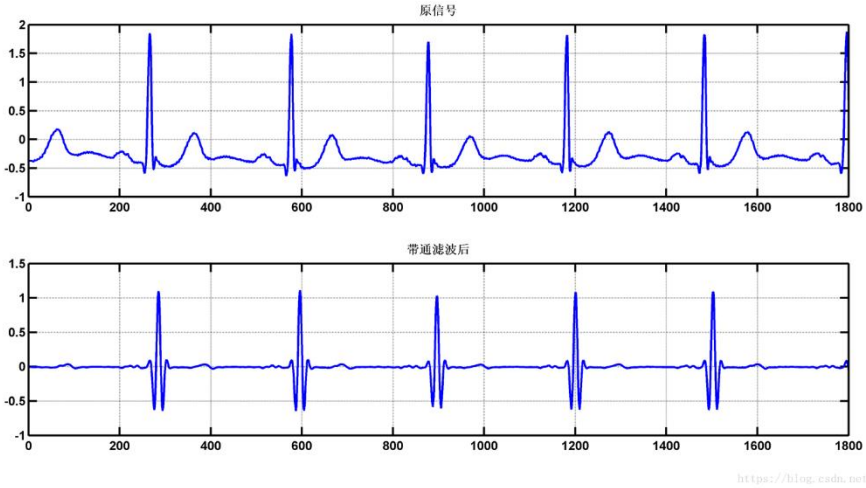
3.示例：QRS 波定位算法設計與評估

在 QRS 波識別領域，不得不提的就是經典的 Pan-Tompkins 算法[1]和 Cuiwei Li 等人發表的小波變換法[2]。已經有大神實現並開源了 Pan-Tompkins 算法的 Matlab 代碼，已經放到了本人 github 上：pan_tomkin.m。不過這裡不打算以這兩個算法為例講解，因為這兩個算法對初學者來說比較複雜。下面是以本人原創的一個 QRS 波定位算法進行分析。

1）預處理。這一部分的總體步驟圖如下：

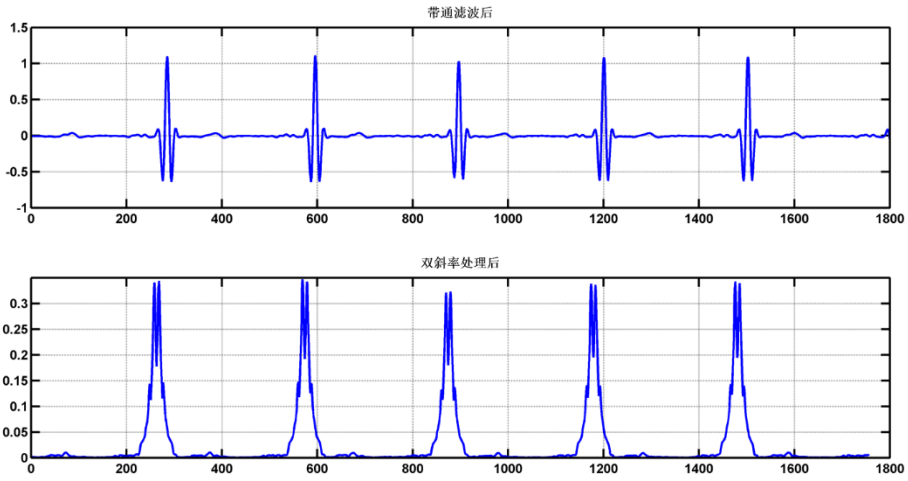


（1）我們對心電信號進行 40 階 FIR 帶通濾波，通帶為 15~25Hz，大致為 QRS 波所在頻段。關於濾波器的設計可以使用 Matlab 的 Fdatool 工具，得到相應的濾波器係數，然後套用 filter 內建函數完成濾波。以數據庫的 103 號信號為例，濾波前後的波形示意如下：

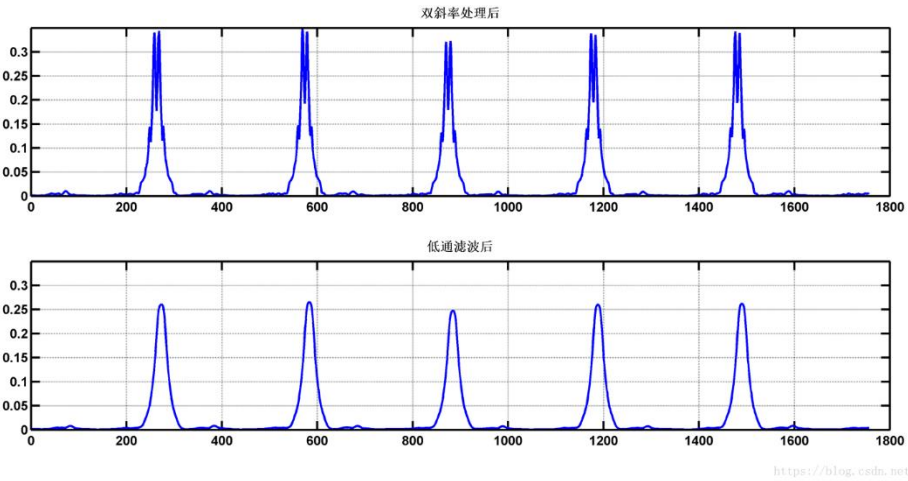


通過對比可以看到，濾波後 P 波，T 波被明顯削弱，體現了上一部分中預處理目的的第（1）點。

（2）對濾波後的波形“雙斜率”預處理。所謂“雙斜率”處理是參考自另一篇文獻[3]。基本思想是分別在一個點的左右兩側的某個區間內尋找最大平均斜率與最小平均斜率，然後分別用左側最大斜率減去右側最小斜率，用右側最大斜率減去左側最小斜率，再求取兩者中的最大者。過程有些複雜，但其實其基本動機就是利用 QRS 波兩側較陡的性質，只有 QRS 波這樣的尖峰在經過上述處理時才會有很大的響應。在這裡，我們設定這個尋找斜率的區間是左右兩側 0.015s~0.060s 處，此為經驗參數。經過處理，可對比前後波形：

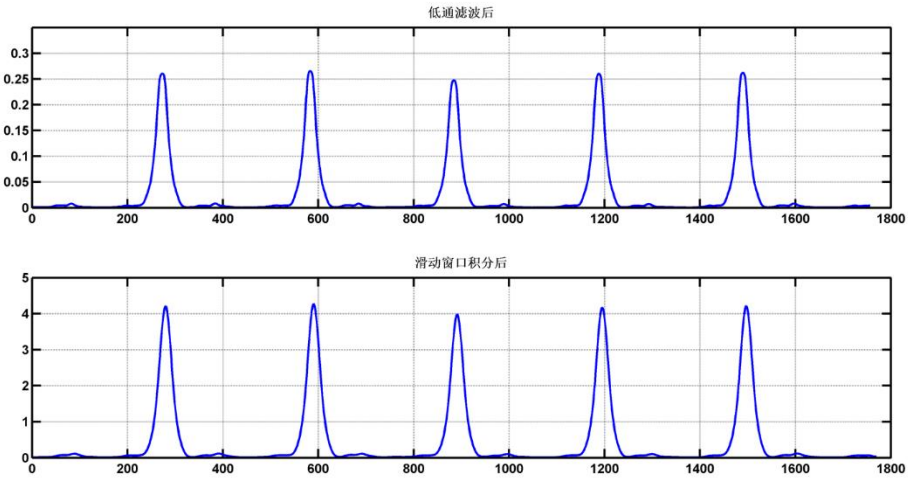


可以看到，雙斜率處理後，波形模式更為單一，體現了預處理目的的第（2）點。但是，波形出現了雙峰現象，一定程度上不利於精準檢測。因此，在雙斜率處理後繼續低通濾波（截止頻率 5Hz，經驗參數），使得波形更光滑：



可以看到，低通濾波後的波形變得非常光滑，雜波基本消失，模式也非常單一，基本上已經達到了進一步閾值處理的要求。

（3）滑動窗口積分。仔細觀察前幾步，每一步得到的波形由於濾波或是求斜率的因素，其幅度值越來越小，而過小的幅值其實不利於檢測。這裡我們利用滑動窗口積分，使得絕對振幅增大，並使波形進一步光滑，滑動窗口寬度設為 17 個採樣點，為經驗參數：



積分後，波形幅值明顯增大（注意縱坐標），體現了預處理目的的第（2）點。至此，預處理階段完成，經過處理後，原始的心電信號變成了一個個模式單一的波峰組成的信號，每個波峰對應一個 QRS 波。相比原信號，預處理後的信號更易定位檢測，可以說達到了預處理的目的。

2)自適應閾值設計

與上述所說的自適應閾值設計原則相契合，這個算法的思路是：

（1）**自適應閾值要跟隨信號實時變化**。問題是如何決策此時閾值的變化。這裡我採用了一種常用的策略：雙閾值。一高一低，當某個波峰超過低閾值時，我們認為檢測到了一個 QRS 波，然後通過比較波峰振幅與高低閾值的關係，調整閾值。

（2）**為了保證變化的穩健**，閾值需要根據前面已經檢測到的正確波峰振幅變化，另外設定下限，以確保不會過高或過低。

通過以上兩點，我設計自適應機制如下（THR1 為高閾值，THR2 為低閾值）：

$$THR1=\begin{cases}0.7*mean(peak\_buffer), if\ (peak>THR1);\\ THR1-\dfrac{|peak-mean(peak\_buffer)|}{2}, if\ (THR0<peak<THR1);\\ THR1\_\lim, if\ (THR1\le THR1\_\lim).\end{cases}$$

$$THR0=\begin{cases}0.25*mean(peak\_buffer), if\ (peak>THR1);\\ 0.4*peak, if\ (THR0<peak<THR1);\\ THR0\_\lim, if\ (THR0\le THR0\_\lim).\end{cases}$$

其中，peak 指檢測到的峰值，peak_buffer 指存儲了當前峰值之前 8 個峰值，mean(.)則是求均值。THR1_lim 與 THR2_lim 為兩個經驗常數，代表兩個閾值變化的下限，在這里分別取值為 0.3 和 0.23。

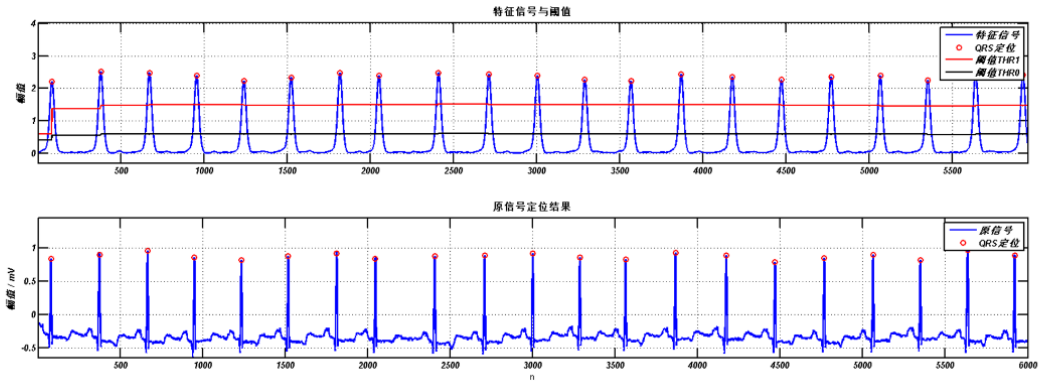
具體來說，噹噹前波峰高於高閾值時，我們傾向於認為此時閾值偏低，閾值變化為之前檢測到的 8 個波峰平均值的 0.7 倍，0.25 倍（均為經驗係數），如果相比最近的 8 個峰值，閾值確實偏低程度很大，則就算乘上了一個小於 1 的係數（例如 0.7，0.25），自適應閾值還是會增大，如果偏低程度不大，則不一定增大。之所以乘上係數還是因為要保持變化的穩健，保證只有閾值明顯偏低時才會明顯增大；噹噹前波峰在高低閾值之間，我們傾向於調低高閾值，而低閾值要與高閾值保持距離，因此也傾向於調低一些，不過我們傾向於認為此時的波形較為異常（因為較小，在兩閾值之間），不宜根據前面的平均峰值變化，因此此時閾值變化與當前波峰的值相關。具體變化機制可以參考上面的公式，體現實時變化原則。另外重複一下，當波峰低於低閾值時，認為是噪聲。

（3）**防漏檢與錯檢。**其實本身雙閾值的設計一定程度上就可以防漏檢。因為一高一低的閾值，相比只有一個閾值，可以捕捉到更多層次的波峰。另外在防錯檢方面，我採用了“不應期”的方式，當兩個波峰過於靠近時，只取較大的波峰。這個“過近”指的是距離低於 0.24s，即不應期的長度；雙閾值的變化存在下限，也防止了一些噪聲被錯檢為 QRS 波。

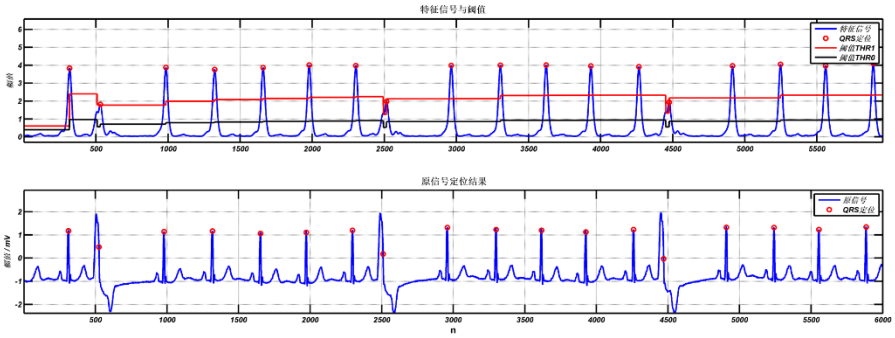
另外，雙閾值的初始值是被提前設定好的經驗係數，與其他經驗係數一樣，一旦確定就不會因記錄的不同而變化。體現了上述自適應閾值設計的第（1）原則。

以上閾值的操作，均在原信號預處理後的特徵信號上進行，檢測到的波峰都認為對應於原信號中的一個 QRS 波。另外預處理中的兩次濾波都造成了信號的延遲，我們檢測到特徵信號的波峰後，可減去延遲，即可得到原信號中的 QRS 波的位置。

說了這麼多確實有些抽象，還是看一下直觀的圖：



上圖這是以 100 號信號為例。由於本身信號比較穩定，因此閾值的變化主要在信號開始。可以看到初始時閾值明顯偏低，自適應閾值很快向上變化找到了合適的值。下面再看一條變化比較大的信號 119：



可以看出雙閾值可以很好地跟隨信號變化，從而正確檢出 QRS 波的位置。注意,這裡檢測的"正確"並非是指精準地落在 QRS 波的波峰或波谷中,儘管精確的人工標註往往就是在波峰或波谷上.美國 EC38 標準提供了一個評判標準:與人工標準差距在 150ms 內,即可認為定位成功.事實上,這個標準相當寬鬆,以至於現在的 QRS 波定位算法的準確率普遍在 99%以上(以 MIT-BIH 數據庫為測試數據), 而一些出色的算法甚至達到了 99.9%以上.
以上原創算法已開源,放在本人的 github 上: DS_detect.m, 已封裝為函數。以上的算法可能理解起來相當抽象，建議結合代碼（有註釋）體會。

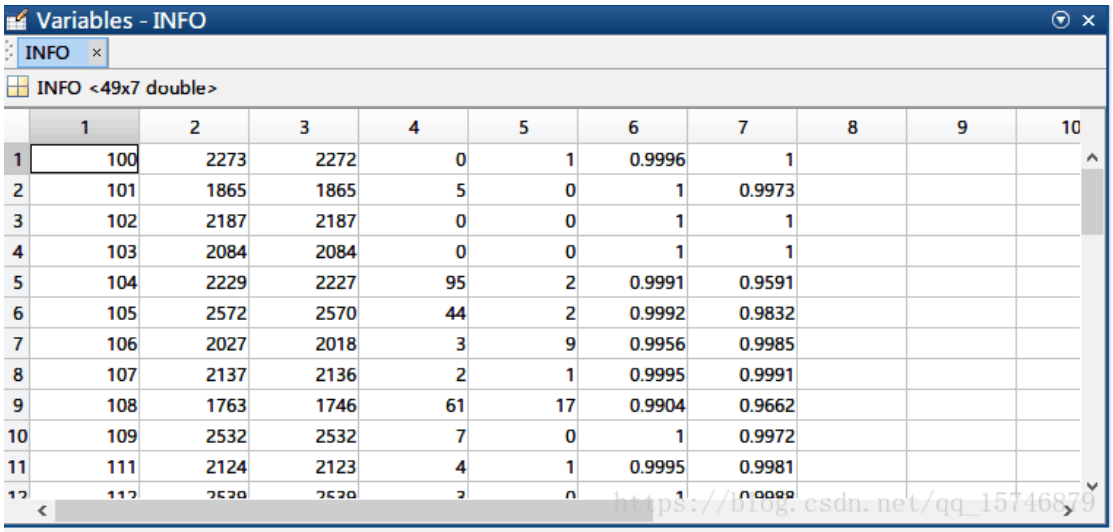
3）算法評估
上一部分中曾說過，目前 QRS 波定位檢測算法的“準確率”已經普遍在 99%以上。但這麼說其實不准確，因為衡量 QRS 波檢測算法的指標並非我們印像中的準確率，而是有兩個常用指標：敏感度（SE）和正預測率（P+），定義如下：

$$\begin{cases} SE = \dfrac{TP}{TP+FN} \\ P+ = \dfrac{TP}{TP+FP} \end{cases}$$

其中 TP 代表檢測正確的心拍個數，FN 代表漏檢的心拍個數，FP 代表錯檢的心拍個數。我們嘗試對整個 MIT-BIH 數據庫跑一遍，統計算法

的性能參數。這一部分不是算法的核心，所以在這裡不再講具體的實現步驟，不過已經開源了測試代碼:DS_test.m,為 Matlab 腳本文件。在運行之前註意數據庫路徑的設置。注意在代碼的第 164 行，調用了上面的算法 DS_detect。

DS_detect 和 DS_test 最好在一個文件夾中，或者將這些文件放到 Matlab 可以檢索到的目錄中。以上都沒有問題後，直接運行 DS_test，等待其完成。當然也可以把 DS_detect 改為其他算法，只要搞清楚其輸出哪個是 QRS 波的索引就可以。DS_test 程序完成後，在工作空間中可以找到名為 INFO 的變量，雙擊打開：



	1	2	3	4	5	6	7	8	9	10
1	100	2273	2272	0	1	0.9996	1			
2	101	1865	1865	5	0	1	0.9973			
3	102	2187	2187	0	0	1	1			
4	103	2084	2084	0	0	1	1			
5	104	2229	2227	95	2	0.9991	0.9591			
6	105	2572	2570	44	2	0.9992	0.9832			
7	106	2027	2018	3	9	0.9956	0.9985			
8	107	2137	2136	2	1	0.9995	0.9991			
9	108	1763	1746	61	17	0.9904	0.9662			
10	109	2532	2532	7	0	1	0.9972			
11	111	2124	2123	4	1	0.9995	0.9981			

從左到右，第一列：記錄編號，第二列：人工標記心拍數，第三列：算法正確檢出心拍數，第四列：算法錯檢心拍數，第五列：算法漏檢心拍數，第六列：Se，第七列：P+。看最後一行：

48	234	2753	2751	0	2	0.9993	1
49	0	109966	109586	648	380	0.9965	0.9941
50							
51							
52							

可以看出算法共錯檢 648 個心拍，漏檢 380 個心拍，總體 Se 達到了 99.65%，P+達到了 99.41%，匯總如下：

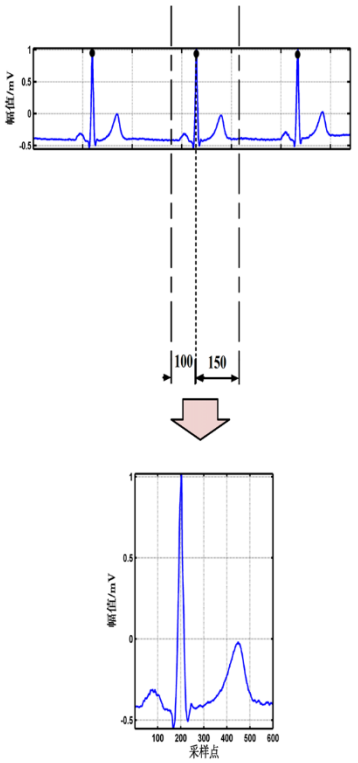
SE	P+	软件环境	运行时间/记录
99.65%	99.41%	Intel Core i5-3230M @2.6GHz, 8GB RAM MATLAB R2013a	~14s

在我的筆記本電腦上，該算法處理一條 30min，採樣率 360Hz 的記錄需要 14s 左右。各項指標雖達不到頂尖，但也不算差，算是有一定的可用度了。接下來，我們將使用該算法將我們需要的心拍截取出來。

4.心拍截取

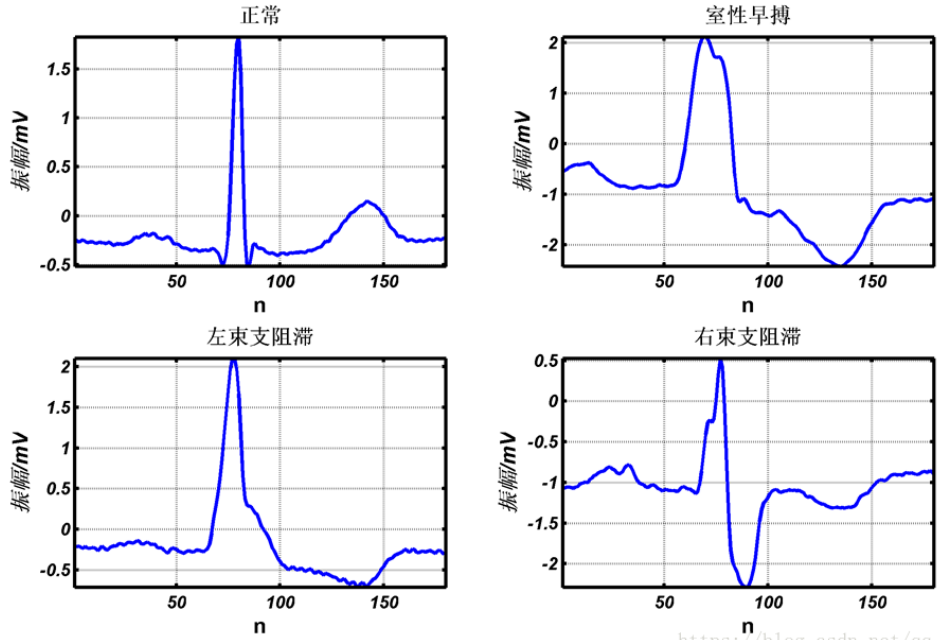
這裡其實有一個矛盾。無論我們的 QRS 波定位算法有多厲害，都很難 100%將人工標註過的所有心拍定位準確，必然有錯檢和漏檢。這就導致我們無法 100%利用好人工標記的心拍。這裡我們採用一個折衷的手段：忽略漏檢的心拍，而錯檢和正確檢出的心拍都截取出來。由於 QRS 定位算法的、性能已經夠好了，所以截取出來的心拍絕大多數是正確檢出的心拍，只有少部分為錯檢心拍。而我們接下來所要使用的機器/深度學習算法都具有一定的魯棒性，較少的錯檢心拍數不會對結果產生太大影響。至於心拍的類型（標籤），我們以距離該心拍最近的人工標記的類型為準。

另外，我們截取心拍其實不是精確截取，即從 P 波起點到 T 波終點，而是採取一種較為粗略的手段：以 QRS 波的位置為基準，分別向前向後包括若干點，然後將這一段數據點截取出來作為心拍。如下圖所示為什麼不精確截取呢？是因為檢測 P 波和 T 波及其起始點或終點非常困難，目前也沒有一個性能足夠好的算法滿足應用需求。所以，大部分文獻採用的是這種粗略截取的方法，只不過前後包括的點數有區別：



這裡我們向左包含 100 個點，向右包含 150 個點，即截取的每個心拍長度為 250 個點（約 0.7s）這一部分的實現代碼也已開源：

SegBeat.m 為腳本文件。同樣，數據庫路徑，代碼文件存放位置要正確。該代碼與 DS_test.m 有類似之處，也調用了 DS_detect 算法進行 QRS 檢測（第 164 行，同樣也可以置換為其他 QRS 波定位算法）。至於其他實現細節，作為初學者可不必了解，因為確實很繞，並且也並非是我們的主線任務。運行完成後（時間可能會比較長），在工作空間可以找到 4 個變量，Nb,Lb,Rb,Vb,為我們接下來的目標類型心拍數據集合，分別為“正常（N）”，“左束支阻滯（LBBB）”，“右束支阻滯（RBBB）”，“室性早搏（PVC）”，示例如下（可能橫坐標表示有些不准確，請忽略。。。。）：



實際上，同一類型的心拍間變化非常大，並非只是示例的四個心拍那麼簡單，這也就是我們為什麼後面要使用高大上的機器/深度學習算法來分辨。我們以.mat 的形式將這四個變量保存（N_dat.mat,L_dat.mat,R_dat.mat,V_dat.mat），它們各自包含了一種類型的心拍，是我們接下來使用機器/深度學習算法要進行分類識別的目標。至此，我們所需要的識別對像已經成功準備好了。

5.小結

本節內容很多，也很抽象。實際上，QRS 波檢測定位算法的設計很大程度都要靠經驗，只不過個人認為有了上面說的幾條原則的指導下，方向可以更明確。相關代碼已經開源,大家可以結合代碼來加深理解,多上手,多嘗試.把目標心拍截取之後,我們接下來就可以進行機器/深度學習的應用了.

ECG ×AI: 機器/深度學習的 ECG 應用入門 (4)

傳統機器學習:特徵工程+分類器

1.引言

經過前面的工作，我們已經解決了數據來源和識別對象問題。那麼接下來，我們就要進行機器/深度學習算法的應用了。由於本人寫這些博文的目的不在於講解機器/深度學習理論，所以，涉及到這部分的內容基本不會做太深的原理探討，不懂的請自行百度，網上的資料有的是。
這一部分，我們先從傳統機器學習框架開始，幾乎所有利用傳統機器學習算法進行分類的問題都遵循“特徵工程+分類器”的思路，即先人工構造若干能體現樣本特性的特徵，然後後端選擇分類器，例如支持向量機（SVM），k 最近鄰分類器（k-NN），決策樹，貝葉斯分類器等等單個或集成的算法。然後，再不斷進行優化，包括特徵的選優和分類器參數的調整，達到理想的效果。一般說來，後端的分類器算法已經成型，想實質改進它們需要很深的理論基礎，在應用領域很少對它們進行大的改動。所以，我們已經清楚了，若要採用傳統機器學習算法框架，設計好特徵是關鍵。那麼對於我們的 ECG 信號來說，又該怎樣設計特徵呢？

2.特徵設計與分類

1）對於傳統機器學習，不設計特徵行不行？
我們之前已經提取了心拍，每個長度為 250 個採樣點，那麼能不能直接把這些採樣點當作特徵呢？
乍一想，這麼做好像也沒什麼問題，還能省下特徵提取的步驟。但其實這麼做確實不是個好的選擇，如果我們不做任何處理，那麼特徵維度就是特徵點個數，為 250 維。對於這麼多的特徵，機器學習算法往往都會遭遇“維數災難”問題（機器學習領域經典理論，可自行查閱相關資料），從而達不到有效學習的目的。況且，這些特徵點的冗餘度很大，除去信號的快變區域，相鄰的幾個點

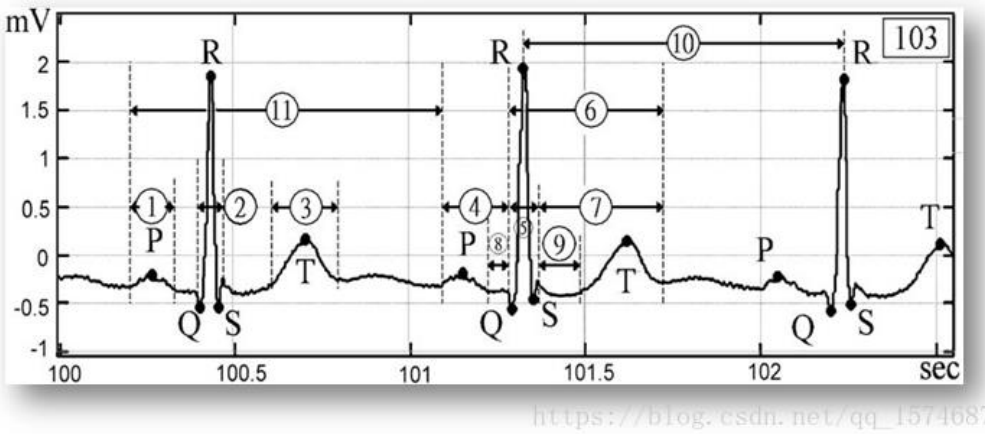
表徵的信息幾乎沒有什麼區別，這樣做無疑也會加重分類器的負擔。因此，對於 ECG 信號，設計有效的特徵對於後端的分類來說，是很有必要的。

2）常用 ECG 特徵

（1）ECG 形態特徵

通過查閱一些 ECG 領域的醫用手冊，我們可以知道，醫生在通過 ECG 診斷心血管疾病時，其實關注的是各個波形的變化情況，例如，當 QRS 波變大變寬時，可能發生了室性早搏；ST 段抬高時，可能發生了心肌梗死。這樣，通過最直觀的波形變化，結合醫生的經驗，可以進行疾病的診斷。所以，這就引出來了第一類常用的特徵：形態特徵。常用的形態特徵有：

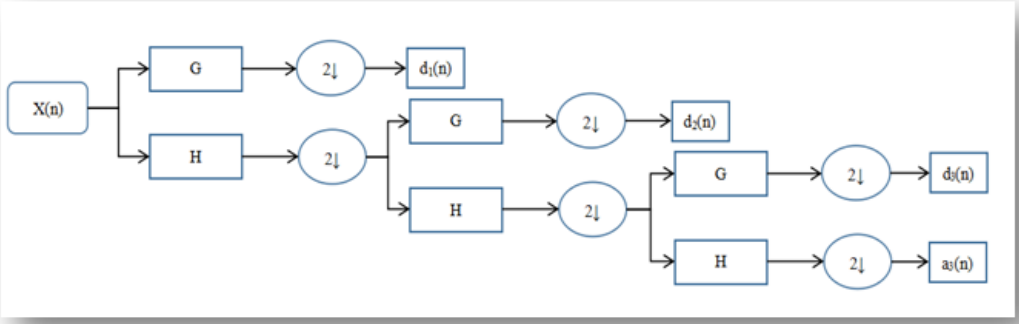
① P 波振幅②QRS 波振幅③T 波振幅④PR 間期⑤QRS 間期⑥QT 間期⑦ST 間期⑧PR 段水平⑨ST 段水平⑩RR 間期，如下圖所示：



以上這些都是可以直接從 ECG 信號中提取到的“一級”特徵，由這些特徵，還可以組合為各種豐富的“二級”特徵，例如例如 QRS 波面積（QRS 間期×QRS 振幅）等。對於形態特徵，它的優點在於直觀，可解釋性強，但是缺點也很明顯。那就是需要對心拍的各個波段進行精細的定位，而做到這一點是相當困難的，前面已經說過，目前除了 QRS 波定位檢測算法已經足夠成熟可靠外，其他波段的定位算法可靠性都不高。如果不能對這些波段進行精准定位，那對於後端分類器的影響是很大的。因此，在近幾年的文獻中，單純使用形態特徵的論文已經不多見了。

（2）ECG 變換係數特徵

上面直接提取形態特徵的方法雖然直接，但存在嚴重問題。那能不能通過一些間接的方式得到有效的，較少的，能夠有效表示心拍的特徵呢？目前大多數採用傳統機器學習框架的論文都採用了這樣一種方法：使用一些數學變換處理 ECG，得到較少的係數，用這些係數來表徵心拍。最常見的就是小波變換。利用小波變換能提取多尺度特徵的特性，得到有效的小波係數，來表徵心拍：



小波變換的基礎理論相當複雜，涉及到了大量難懂的數學概念和運算。不過從應用角度來說，我們可以把小波變化看作一個個級聯的低通和高通濾波器，然後還有下採樣操作。上圖中，G 和 H 分別表示不同的高通和低通濾波器，後接 2 倍下採樣，a 和 d 表示了不同尺度下的近似係數和細節係數，這些係數是跟 ECG 信號的內在特點相關的，通過直接使用或進一步處理這些係數，我們可以得到豐富的特徵用於分類。

另外，還有部分文獻採用了多項式擬合的思路。即對一個心拍進行多項式擬合，得到各階的係數作為特徵。總體來說，這些使用 ECG 變換係數的特徵，避免了對一些特徵點的直接定位，從提取難度上來說小了很多。但是，付出的代價是特徵的含義不再直觀，這使得對於這些係數特徵進行選擇時變得困難，往往比較盲目。

以上兩種常用特徵各有利弊，不過從目前的趨勢來看，提取難度更小的第二種特徵更受青睞。而且這些特徵基於數學變換，也具有更多的理論價值和研究價值。這可能是目前此類論文較多的原因。

3）常用分類器

這一點其實不用多說了。基本上所有經典的分類器都可以用，只要能夠得出好的效果。而目前各種分類器算法的工具包支持也很完善了，如果從應用角度來看，我們可以直接套用高效率的工具包，而不用自己重造輪子。廢話不多說，下面直接開始示例。

3.示例：心律失常的 SVM 識別

這一部分中，我們採用大名鼎鼎的支持向量機 SVM 作為後端的分類器。仍然是在 Matlab 環境下進行實驗，相應的工具包使用了著名的 libsvm。由於 libsvm 不是 Matlab 自帶的工具包，其安裝過程略微複雜，需要外部 C++編譯器的輔助。但是這裡我直接將編譯好的工具包共享出來，放在本人的 github 上:libsvm-3.21.zip。解壓後，這個工具包是一個文件夾（libsvm-3.21），想在 Matlab 中使用，需要在 Matlab 中設置路徑（Set path），將這個文件夾中名為

“matlab”子文件夾添加到 Matlab 可檢索的目錄中，就可以調用了。建議將 libsvm 文件夾放在 Matlab 安裝目錄下的 toolbox 文件夾中。另外，由於我是在 64 位平台上編譯的，因此這個工具包只能在 64 位版本上運行。如果是 32 位，可以參考書籍《MATLAB 神經網絡 43 個案例分析》中有關 SVM 的章節，裡面有比較詳細的安裝過程。

在進行以下步驟之前，需要使用 load 函數把我們之前存儲的心拍數據（.mat）導入到工作空間中.這時我們可以發現,正常類的心拍遠遠多於其他三類非正常心拍, 即存在數據分佈不平衡問題.這裡我們為了減少數據對之後分類器訓練和測試的影響,每一類只選取 5000 個進行實驗,也就是數據集規模為 20000.以下所有步驟的 Matlab 代碼可在本人 github 上下載到：Classification-SVM.m,可對照代碼理解並上手.

1)特徵提取

。這裡我們採用小波變換的係數作為我們的特徵. 原因除了在於小波變換的多分辨率屬性外,還在於小波變換中還包含了下採樣操作,可以理解為內嵌了降維的過程.從實現角度來講,Matlab 平台中提供了相應的內建函數供我們使用,大大降低了我們的實現難度.

具體的,我們對每個心拍進行 5 階的小波分解,小波函數利用 db6 小波,使用 Matlab 中的 wavedec 函數（函數具體功能建議查看 Matlab 幫助文檔）可以輕鬆完成:

```
[C,L]=wavedec(sig,5,'db6');
```

其中，C 包含了各階小波變換後的係數，sig 在這裡表示我們所提取的 250 點心拍。經過 5 階小波分解和 2 倍下採樣後，我們取小波變換係數中原信號的“近似”係數，即 5 階分解後的 a 係數，根據 L 中的信息可以知道是前 25 個點。提取“近似”係數的動機是這些係數去除了一些細節，對更一般性的規律體現更好。這一點在疾病的診斷中非常關鍵，因為不同的病人或相同的病人在不同時間，由於身體狀況的改變可能造成 ECG 的細節變化。這些細節變化是帶有特殊性的，往往不可以作為疾病的一般性特徵。注重一般性規律，是有利於防止機器學習中的“過擬合”問題的。因此，我們這裡選取這 25 個係數為代表每個心拍的特徵。

從這裡看起來，提取特徵貌似也沒什麼難度。但是，我們這裡是最簡單的示例，不一定保證這些特徵就是最優的。真正好的特徵需要非常多的經驗和實驗，同樣以小波變換係數為例，分解階數，特徵個數，小波函數等等的選擇，都會影響特徵的質量。並且這只是直接提取的係數，在此基礎上還可以做各種處理，形成二級特徵。而這些因素在真正的研究中都需要認真考慮和探究。

2）數據集選取與劃分

隨機劃分訓練集和測試集,各佔一半,即訓練集和測試集各有 10000 個樣本. 隨機選取可通過 Matlab 內建 randperm 函數隨機打亂樣本索引,然後截取前 10000 個索引對應的樣本作為訓練集,剩餘的作為測試集來實現.

3）特徵歸一化

在機器學習任務中，特徵歸一化不是必須的步驟。這里為了加快 SVM 收斂，進行了特徵歸一化。這裡的歸一化指的是同一特徵在不同樣本上取值的歸一化,借助 Matlab 內建歸一化函數 mapminmax 實現. 不過這裡要注意的是 mapminmax 函數對於歸一化對象,即樣本特徵矩陣的行列含義的要求,必要時要對矩陣進行轉置. 實際操作時,先使用 mapminmax 函數的正常模式對訓練集特徵歸一化到 0,1 之間,得到歸一化後的訓練集和歸一化信息; 然後使用 mapminmax 的 apply 模式,將訓練集歸一化得到的歸一化信息應用至測試集,完成測試集的歸一化:

```
[train_x,ps]=mapminmax(train_x',0,1); %利用 mapminmax 內建函數特徵歸一化到 0，1 之間；
test_x=mapminmax('apply',test_x',ps);
```

4)模型訓練與測試

調用 libsvmtrain 函數和 libsvmpredict 函數訓練和測試模型,注意兩個函數對於數據順序和格式的要求:

```
model=libsvmtrain(train_y,train_x,'-c 2 -g 1'); %模型訓練；
[ptest,~,~]=libsvmpredict(test_y,test_x,model); %模型預測；
```

libsvm 訓練的默認核函數為 RBF 核函數,需要人為設定兩個超參數 c 和 g. 這裡我們設定為 2 和 1. 注意,不同取值的 c 和 g 可能會導致差異比較大的結果.如果想要取得更好的效果,必須進行"調參",減輕欠擬合與過擬合問題. 而這一部分也是所有機器/深度學習問題中最依賴經驗和最耗費時間的. 常用的調參方法除了手動調整,還有網格搜索,啟發式尋優等方法.

5)結果統計與展示

這一部分主要是根據分類器的實際輸出結果(上一步中 libsvmpredict 輸出的 ptest 向量),與期望結果(標籤)對比, 統計各類別的準確率等.詳細的代碼可參見本人 github. 例如,根據以上的特徵提取和參數選擇, 我們可以得到如下關於準確率的信息(由於這裡每次會選取不同的訓練和測試樣本,因此每次運行結果會有所不同):

```
Accuracy = 96.69% (9669/10000) (classification)
Accuracy_N = 99.68%
Accuracy_V = 90.90%
Accuracy_R = 97.58%
Accuracy_L = 98.49%
```

在本次實驗中,我們的 SVM 模型總體預測準確率為 96.69%, 其中四類目標類型的準確率分別為正常(N):99.68%, 室性早搏(V): 90.90%, 右束支阻滯（R）:97.58%, 左束支阻滯（L）:98.49%. 代碼中其實還給出了一種常見的衡量多分類的方法：混淆矩陣。具體定義參見：https://baike.baidu.com/item/混淆矩陣/10087822?fr=aladdin 工作空間中打開名為“Conf_Mat”的矩陣，可見：

Conf_Mat <4x4 double>

	1	2	3	4
1	2517	6	0	2
2	15	2248	82	128
3	28	13	2419	19
4	3	35	0	2485

類別 1，2，3，4 分別表示 N，V，R，L。從混淆矩陣中，我們可以看出預測結果的分佈情況。對角線上是各類別正確預測的個數，其餘均為錯誤預測個數。例如，第 2 行第 4 列，有 128 個早搏心拍被錯誤預測為左束支阻滯，是所有錯誤中最多的。可以猜測，早搏心拍與左束支阻滯心拍可能在表現上更相似一些；而第 4 行第 3 列錯誤預測數為 0，即沒有左束支阻滯心拍被錯誤預測為右束支阻滯，說明可能兩者差異更大一些。通過分析混淆矩陣，可以對分類器的性能和數據分佈做更精細的分析，根據經驗採取相應措施優化。

綜上，我們以經典的 SVM 為例，完成了一個傳統機器學習框架的最基本 ECG 應用。從特徵提取到模型訓練與評估，最終得到了一個看起來還不錯的效果（這裡還存在一些問題，之後的內容中會有所提及）。以上強烈建議結合代碼理解和上手。當然，這裡面還大有文章可做，包括怎麼樣提取質量更高的特徵，怎麼樣選取更優的超參數，以及選取更適合的分類器等等。但萬變不離其宗，總體上，在傳統機器學習框架中，我們主要的精力應該還是集中於“好”特徵和參數的選擇。

4.小結

這一部分將傳統機器學習框架的 ECG 分類應用大致走了一遍流程。看似簡單有效，但其實特徵的選取和選擇是一件很困難的事情。想要設計出好的特徵，必須具有相當的經驗和大量的實驗。下面推薦了幾篇文獻，可以更進一步體會。

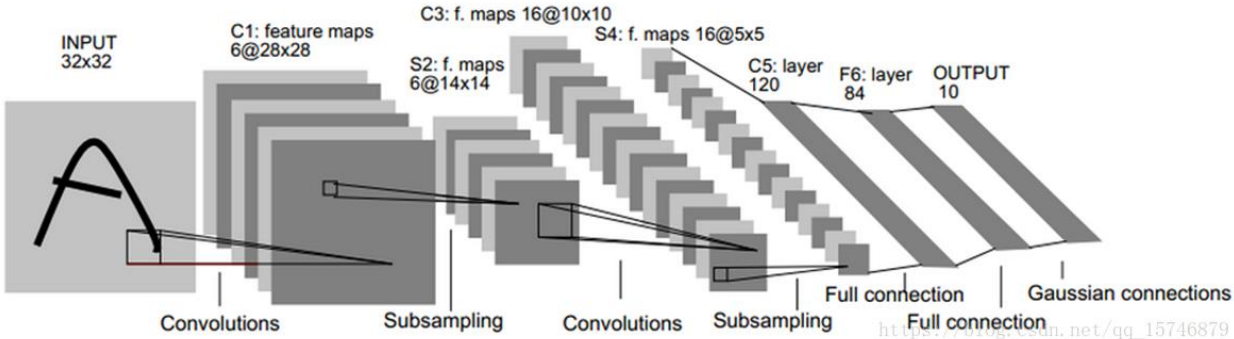
ECG ×AI: 機器/深度學習的 ECG 應用入門

(5)

深度學習：卷積神經網絡（CNN）

1.引言

上一部分簡單介紹了傳統機器學習框架在 ECG 分類領域的基本應用。傳統機器學習框架對於人工特徵非常依賴，如果算法設計者沒有足夠經驗，很難提取出高質量的特徵，這也是傳統機器學習框架的局限性。近幾年來以卷積神經網絡（Convolutional Neural Network，CNN）為代表的深度學習技術蓬勃興起，其優勢在於可以從大數據中自動習得特徵而無需人工設計特徵，其在多種任務，例如圖像分類與定位，語音識別等領域都展現出了十分強大的性能。這一部分講解如何使用 CNN 來完成與上一節同樣的任務。如果不熟悉 CNN，網上有很多的資料可以參考。



2.CNN 可用的原因

根據大量的文獻，我們可以知道，CNN 非常適合處理圖像，以至於基於傳統機器學習的算法完全不能相比。圖像這樣的數據形式，存在局部與整體的關係，由低層次特徵經組合可形成高層次特徵，並可以得到不同特徵間的空間相關性。仔細想想，我們的 ECG 信號似乎也存在這樣的特性，局部的一些波形與整體結果息息相關，而診斷，實質上就是由一些低層次的，可見的波形變化抽象成一些疾病的概念，即高層次的特徵。而波形的空間關係也往往蘊含了豐富的信息。這樣看起來，ECG 與圖像有很大的相似之處，而 CNN 通過以下幾點可有效利用上述特點：

1）局部連接：CNN 網絡可提取數據局部特徵。

2）權值共享：大大降低了網絡訓練難度，每個卷積層有多個 filter,可提取多種特徵。

3）池化操作和多層次結構：實現數據的降維，將低層次局部特徵組合無為較高層次特徵。
由於我們處理的 ECG 心拍是一維信號，因此需要使用 1 維 CNN 處理。

3.示例：1 維 CNN 的 ECG 心拍分類

該部分的 CNN 內部代碼改編自：
https://github.com/rasmusbergpalm/DeepLearnToolbox，以 Matlab 寫成，目前放在了本人 github 上。而 CNN 的內部代碼比較複雜，初學者很難短時間內看懂領會。因此這一部分主要講解如何使用。如果對代碼的具體實現感興趣，可以參照 https://blog.csdn.net/zouxy09/article/details/9993743 關於 2 維版本 CNN 的代碼解釋，本部分中所使用的 1 維 CNN 代碼架構與其基本相同。
與前面一樣，解壓 1DCNN.zip 將名為“1DCNN”的文件夾複製到 Matlab 安裝目錄 toolbox 文件夾下，然後在 Matlab 中設置路徑（Set path），將 1DCNN 文件夾添加到 Matlab 可檢索的目錄中，就可以調用了。前幾步，有關數據的載入，標籤的生成，數據的劃分和處理等與前一部分一致，但是，這裡我們不再提取特徵。而是要搭建 CNN 結構，建議參照放在 github 上該部分的代碼：Classification-CNN.m。
1）CNN 結構設計
直接上代碼：

```
cnn.layers = {
    struct('type', 'i') %input layer
    struct('type', 'c', 'outputmaps', 4,
'kernelsize', 31,'actv','relu') %convolution layer
    struct('type', 's', 'scale', 5,'pool','mean')
%sub sampling layer
    struct('type', 'c', 'outputmaps', 8,
'kernelsize', 6,'actv','relu') %convolution layer
    struct('type', 's', 'scale', 3,'pool','mean')
%subsampling layer
};
cnn.output = 'softmax'; %確定 cnn 結構；
```

這裡，我們要設計一個具有 2 個卷積層，2 個池化層，和 1 個全連接層的 1 維 CNN。這個 CNN 工具箱比較簡單，目前只支持卷積層和池化層互相交錯放置。網絡層的類型由'type'指定，後接'i'，'c'，'s'分別指輸入層，卷積層和池化

層。“outputmaps”設定該層有多少輸出特徵圖，從而決定卷積核的數目；
“kernelsize”後的數字指定 1 維卷積核的大小；“scale”後數字指定池化步長；
“pool”後的字符串指定池化類型，“mean”：平均池化，“max”：最大池化；
“actv”後的字符串指定激活函數類型，“relu”：ReLU，“sigm”：Sigmoid，
“ tanh”：Tanh。cnn.output="softmax"指後面的全連接層要進行 softmax 操作，另外還可以指定為“sigm”：使用 sigmoid 函數映射，“linear”：線性映射。
以上這些參數或選擇需要人工指定，可自行模仿調整。另外值得一提的是，這個工具箱比較簡單，因此限制也比較多：
（1）只支持卷積層和池化層互相交錯放置。（2）卷積層不會對特徵圖外圍進行零填充，也就是每次卷積後特徵圖長度會減少 kernelsize-1.（3）池化層的每一步池化區域不重疊，即池化區域長度等於步長。（4）池化層的輸入特徵圖長度必須能被池化步長整除。
因此參數的調整必須考慮和滿足以上的限制，否則就會報錯。

2）超參數的確定

```
%確定超參數；
opts.alpha = 0.01; %學習率；
opts.batchsize = 16; %batch 塊大小；
opts.numepochs = 30; %迭代 epoch
```

接下來我們要使用隨機梯度下降法（SGD）法訓練，因此需要指定學習率，batch 塊大小，以及需要迭代的 epoch 數。這些參數的取值，尤其是學習率，可能會對結果產生較大的影響。
3）建立，訓練，測試 CNN

```
cnn = cnnsetup1d(cnn, train_x, train_y); %建立 1D CNN;
cnn = cnntrain1d(cnn, train_x, train_y,opts); %訓練 1D CNN;
[er,bad,out] = cnntest1d(cnn, test_x, test_y);%測試 1D CNN;
```

cnnsetup1d 會根據我們前面設定的 cnn 結構和訓練數據的結構生成一個 CNN 網絡；cnntrain1d 會使用訓練數據和我們設定的學習率等參數訓練已經建立好的網絡，這一步也是最耗時的；cnntest1d 會測試訓練好的網絡，輸出錯誤率（er），分錯的樣本索引（bad），以及實際的輸出值（out）。工作空間中，名為 cnn 的結構體保存了所有的權值和偏置等關鍵信息，內涵多級結構體，可雙擊打開查看。也可保存為.mat 文件以備後續使用。
4）結果統計與展示

這一步的流程與上一節傳統框架實驗相同，可參照開源代碼。根據我們設定的參數進行訓練和測試，得到的結果：

```
Accuracy = 98.79%
Accuracy_N = 99.80%
Accuracy_V = 97.17%
Accuracy_R = 99.45%
Accuracy_L = 98.75%
```

使用 CNN（總體準確率 98.79%）貌似得到了比人工特徵+SVM（總體準確率 96.69%）更好的結果。當然兩者還都有優化空間，尤其是在超參數的選取上。混淆矩陣仍然保存於工作空間名為“Conf_Mat”的矩陣中，可雙擊打開查看：

Conf_Mat <4x4 double>

	1	2	3	4
1	2447	1	4	0
2	6	2435	36	29
3	0	10	2548	4
4	2	29	0	2449

綜上,我們搭建了 1 維 CNN 實現了心拍的基本分類. 相比傳統機器學習框架,上手更容易,無需顯式提取特徵. 至於缺點,如果你親自運行一下就知道，模型的訓練過程比較耗時,而我們的 CNN 結構也只是簡單的 5 層. 另外，模型中需要確定的超參數也遠多於前面所提及的 SVM,想要調到一個理想狀態也不是太容易,沒有具體的理論指導，需要大量的實驗. 這也是目前深度學習飽受詬病的問題之一.

4.小結

這一部分簡單講述瞭如何使用深度學習算法——1 維 CNN 進行基本的心拍分類。除去特徵提取，其他的數據處理和效果評估與前面一部分基本一致。這裡用到的 CNN 工具箱支持功能不是很全面，只能滿足一些基本需求。如果想在深度學習 CNN 方面深挖，建議使用 Tensorflow，Pytorch 等流行的深度學習框架，支持功能全面，速度更快。不過這就需要一些 Python 及其第三方庫的使用基礎了。目前 1 維 CNN 用於心拍的分類的文章已經比較多了，下面也推薦了幾篇。這一部分涉及到的代碼可以從下面的網址中看到。