UNIVERSITÀ
POLITECNICA
DELLE MARCHE

Corso di Laurea in Ingegneria Informatica e dell'Automazione

OTTIMIZZAZIONE DI ALGORITMI DI FILTRAGGIO PER L'ESTRAZIONE DELLA FREQUENZA CARDIACA DA SEGNALI FOTOPLETISMOGRAFICI

OPTIMIZATION OF FILTERING ALGORITHMS FOR HEART RATE EXTRACTION FROM PHOTOPLETHYSMOGRAPHIC SIGNALS

Relatore:
PAOLO CRIPPA

Tesi di Laurea di:
GIOVANNI BALDASCINO

Correlatore:
GIORGIO BIAGETTI

A.A. 2018/2019

INDICE

1. INTRODUZIONE

1.1 Fotopletismografia

La fotopletismografia (PPG, photoplethysmography) è una tecnica che permette di individuare la frequenza cardiaca (HR, heart rate) usando dei sensori che misurano la quantità di luce riflessa dalla pelle in relazione al flusso sanguigno sottocutaneo.
Come tecnica è fortemente promettente al fine di monitorare l'HR in attività di tutti giorni o anche durante attività sportive, inoltre il PPG è stato ampliamente studiato in applicazioni cliniche per fornire informazioni utili riguardo all'HR o alle sue variazioni.

Il problema principale è l'alta sensibilità al movimento che produce alterazioni del segnale, impedendo spesso un accurato tracciamento della frequenza cardiaca; se il soggetto non è perfettamente immobile, gli artefatti di movimento (MAs, motion artifacts) si sovrappongono alla frequenza cardiaca rendendo la sua estrazione un arduo compito.

Esistono molti approcci diversi per mitigare l'effetto dei MAs; noi ci soffermeremo su uno di questi che è stato presentato nella pubblicazione del Dipartimento di Ingegneria dell'Informazione

dell'Università Politecnica delle Marche: "Reduced Complexity Algorithm for Heart Rate Monitoring from PPG Signals using Automatic Activity Intensify Classifier".

Qui vi è presentato un approccio basato su una separazione dei sottospazi che formano i segnali.

Il segnale PPG viene prima decomposto in vettori singolari per facilitarne la pulizia isolando il sottospazio dei MAs e in seguito un tracker parametrizzato ripulisce ulteriormente la stima istantanea per tracciare l'HR finale.

L'obbiettivo di tale approccio è di ridurre la complessità computazionale per facilitarne l'implementazione su piccoli dispositivi indossabili, quali smartwatch.

Fig. 1

2. ANALISI

2.1 Matlab

Fig. 2

L'algoritmo è interamente sviluppato in Matlab.
MATLAB (abbreviazione di Matrix Laboratory) è un ambiente per il calcolo numerico e l'analisi statistica scritto in C, che comprende anche l'omonimo linguaggio di programmazione creato dalla MathWorks. Matlab consente di manipolare matrici, visualizzare funzioni e dati, implementare algoritmi, creare interfacce utente e interfacciarsi con altri programmi.
L'interfaccia principale di Matlab è composta da diverse finestre che è possibile affiancare, spostare, ridurre a icona, ridimensionamento e così via.
Le finestre principali, più usate, sono quattro:

- *Command Window (prompt dei comandi)*: è una finestra dell'interfaccia principale di Matlab, nella quale è possibile

digitare comandi supportati e visualizzare a schermo in tempo reale i risultati.

- *Workspace*: è lo spazio di lavoro (o spazio di memoria) contenente le variabili dichiarate. La finestra di Workspace elenca tutte le variabili allocare in workspace in questo momento, e dà la possibilità di allocare nuove variabili (ad esempio da un file di testo).

- *Current Directory:* è una finestra che permette, come si può intuire, di esplorare il contenuto delle cartelle sul proprio hard disk; da questa finestra è possibile aprire direttamente file compatibili con Matlab con un semplice doppio click.

- *Command History:* è una finestra in cui sono elencati tutti i comandi digitati di recente, divisi per ora e data; è possibile rilanciare direttamente da Command History un comando digitato in Command Window in precedenza semplicemente con un doppio click.

L'utilizzo di Matlab rimane comodo, inoltre, per l'implementazione e l'utilizzo di algoritmi genetici, argomento che verrà affrontato in seguito.

2.2 Algoritmo di stima della frequenza cardiaca

Il metodo proposto opera sui dati provenienti da due sensori ottici PPG e simultaneamente acquisisce dati da un accelerometro a 3 assi, il tutto montato su polso.
L'algoritmo è composto da tre fasi principali:

1. ***PROCESSING***: dove vengono effettuate le stime dei candidati di artefatto e frequenza cardiaca;
2. ***TRACKING***: dove vengono filtrate le stime dei candidati e, in seguito, smussate per produrre il risultato finale;
3. ***CLASSIFICATION***: dove i parametri necessari per le due precedenti mansioni vengono selezionati in base al tipo di attività in esecuzione.

In figura qui sotto è presente il flowchart dell'algoritmo, si vedono i segnali dell'accelerometro e si prendono due segnali PPG grezzi, relativi allo stesso soggetto; vengono finestrati e in seguito filtrati in modo da poter andare a formare la matrice di Hankel che poi verrà decomposta ai valori singolari (autovalori e autovettori) e infine verrà calcolata la frequenza dominante di ogni autovettore per poter poi rimuovere i MAs e ottenere un segnale utile che, con l'uso degli 8 coefficienti, permette il tracciamento dell'HR.

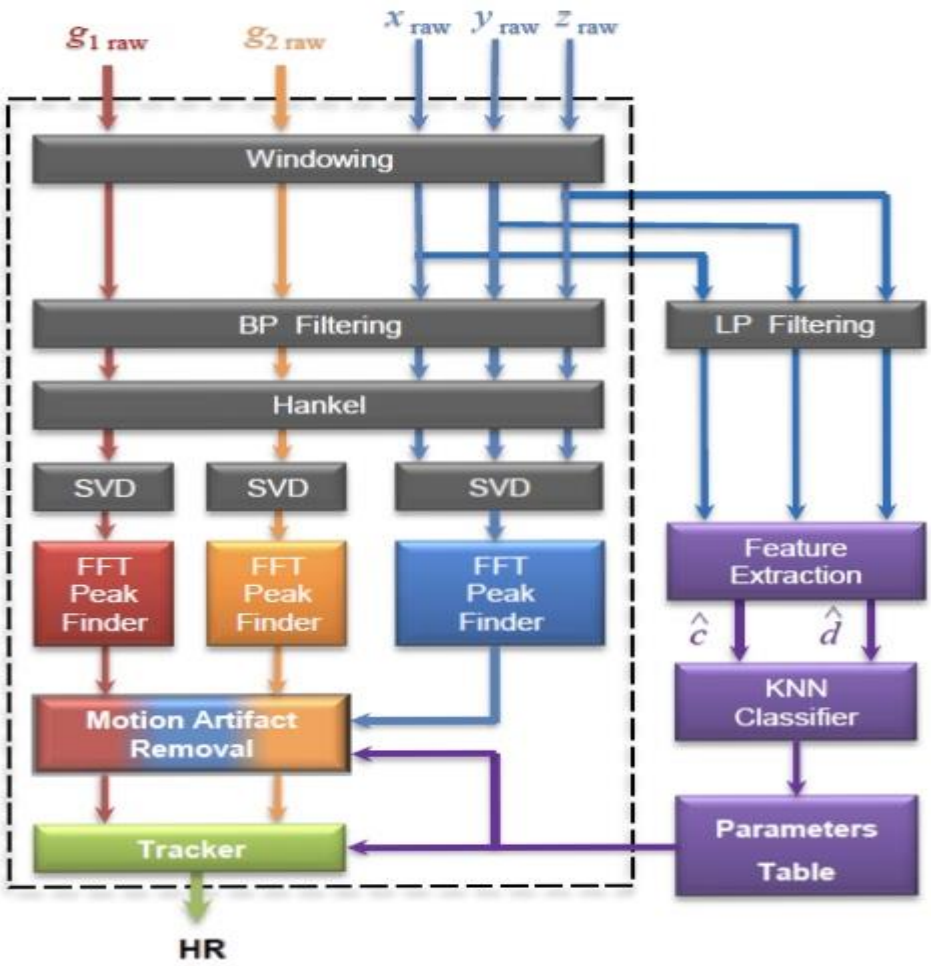

Fig. 3

Senza entrare ulteriormente nel dettaglio gli 8 parametri, per semplicità, li definiamo come segue:

- **ppg_order**: indica il numero di componenti del segnale PPG che si utilizzeranno per ricostruire il segnale utile.
- **maf_order**: indica il numero di componenti del segnale proveniente dall'accelerometro che si utilizzeranno.

- **maf_toler**: indica la distanza entro cui i vettori, post scomposizione, appartengono al sottospazio dei vettori di movimento.
- **alpha**: è un coefficiente utilizzato per pulire la cadenza grezza.
- **d_max**: indica la distanza massima tolletara tra l'attuale stima dell'HR e la precedente stima dell'HR.
- **k_range** / **k_center** / **k_fudge**: sono 3 coefficienti che vengono utilizzati per il calcolo di "**k**" che serve per ridurre l'influenza di possibili valori anomali aggiustando dinamicamente la larghezza di banda del filtro.

Naturalmente è impensabile spiegare in modo accurato o comprendere a pieno questi parametri senza presentare nel dettaglio tutti i procedimenti matematici svolti dall'algoritmo, essendo quest'ultimi presentati già nella documentazione sopra citata, ci limiteremo ad estrapolarli e presentarli semplicemente per completezza

L'efficacia dell'algoritmo è stata dimostrata testandolo su 12 soggetti e le attività fisiche sono state la camminata e la corsa.

Qui sotto è mostrata, su Matlab, la matrice "min_max_starts0" contenete questi coefficienti, in cui la prima colonna è l'estremo

inferiore, la seconda colonna è l'estremo superiore e la terza e la quarta rappresentano il valore iniziale da cui parte l'ottimizzazione.

```
min_max_starts0 = [ ...
    2       20      20      20      ; ... % ppg_order (int)
    2       20      10      10      ; ... % maf_order (int)
    0       10      2.0     2.0     ; ... % maf_toler [BPM]
    0.5     0.99    0.8825  0.8825  ; ... % alpha
    1       20      14      14      ; ... % d_max     [BPM]
    0.01    0.99    0.8     0.8     ; ... % k_range
    0.01    0.99    0.5     0.5     ; ... % k_center
    0.01    0.99    0.12    0.12    ; ... % k_fudge
];
```

Fig. 4

Questi parametri dovranno poi essere ottimizzati in base all'attività svolta dal soggetto che si sta analizzando in modo tale da rendere più preciso il calcolo dell'HR, tutto ciò per sopperire al calo di precisione rispetto agli altri algoritmi che calcolano l'HR, a favore di una bassa complessità computazionale.
L'ottimizzazione di questi 8 parametri avviene mediante l'utilizzo di un algoritmo genetico.

2.3 Algoritmo genetico

Un algoritmo genetico è un algoritmo euristico utilizzato per tentare di risolvere problemi di ottimizzazione per i quali non si conoscono altri algoritmi efficienti di complessità computazionale lineare o polinomiale.
L'aggettivo "genetico", ispirato al principio della selezione naturale di Darwin, deriva dal fatto che l'algoritmo attua meccanismi concettualmente simili a quelli dei processi biochimici scoperti da questa scienza.

In sintesi, gli algoritmi genetici consistono in algoritmi che permettono di valutare diverse soluzioni di partenza (come se fossero diversi individui biologici) e che ricombinandole (analogamente alla riproduzione biologica sessuata) ed introducendo elementi di disordine (analogamente alle mutazioni genetiche casuali) producono nuove soluzioni (nuovi individui) che vengono valutate scegliendo le migliori (selezione ambientale) nel tentativo di convergere verso soluzioni "di ottimo". Ognuna di queste fasi di ricombinazione e selezione si può chiamare generazione come quelle degli esseri viventi

2.4 Terminologia inerente all' algoritmo genetico

- *Funzione di Fitness*: è la funzione che si desidera ottimizzare. Per gli algoritmi di ottimizzazione standard, questa è nota come funzione obiettivo.

- *Individui*: Un individuo è qualsiasi punto a cui è possibile applicare la funzione fitness.

- *Popolazioni e Generazioni:* Una popolazione è un array di individui. Ad esempio, se la dimensione della popolazione è 100 e il numero di variabili nella funzione fitness è 3, si rappresenta la popolazione con una matrice 100 per 3. Lo stesso individuo può apparire più di una volta nella popolazione. Ad ogni iterazione, l'algoritmo genetico esegue una serie di calcoli sulla popolazione attuale per produrre una nuova popolazione. Ogni popolazione successiva è chiamata una nuova generazione.

- *Valore di Fitness e Miglior Valore di Fitness*: Il valore di fitness di un individuo è il valore della funzione di fitness per quell'individuo. Poiché di default viene calcolato il minimo della funzione fitness, il miglior valore fitness per una popolazione è il valore fitness più piccolo per qualsiasi individuo nella popolazione.

- *Genitori e Figli*: Per creare la generazione successiva, l'algoritmo genetico seleziona determinati individui nella popolazione attuale, chiamati genitori, e li utilizza per creare individui nella generazione successiva, chiamati bambini. In genere, è più probabile che l'algoritmo selezioni i genitori con valori di fitness migliori.

3. SVILUPPO

3.1 Presentazione

Dopo aver introdotto e analizzato i requisiti, entriamo in quello che è stato il mio tirocinio nello specifico.
Avendo già spiegato il funzionamento dell'algoritmo tramite l'uso di 8 coefficienti e la loro successiva ottimizzazione, che era avvenuta su 12 soggetti in movimento, dove i movimenti in questione erano la corsa e la camminata; il mio compito è stato l'ottimizzazione di questi coefficienti ma per due attività fisiche diverse, nello specifico il dataset fornitomi era relativo a soggetti sottoposti a squat e alla salita/discesa di uno scalino.

Fig. 5

Fig. 6

3.2 *Dataset Squat/Gradino*

Non essendomi occupato personalmente delle misurazioni, mi è stato fornito un dataset relativo a 3 soggetti diversi (Leonardo, Gloria e Maria che per comodità verranno identificati come 01, 02 e 03) i quali sono state sottoposti a 10 squat e alla salita di un gradino per 10 volte.

Per ogni soggetto, nel concreto, sono stati forniti i seguenti file in formato ***.csv*** ("*Comma-Separated Values*" ossia un formato di file basato su file di testo utilizzato per l'importazione ed esportazione di dati, dove ogni riga della tabella è rappresentata da una linea di testo, che a sua volta è divisa in campi separati dalla virgola, ciascuno dei quali rappresenta un valore):

- **PP**: dove la prima colonna è il tempo e la seconda è il PPG

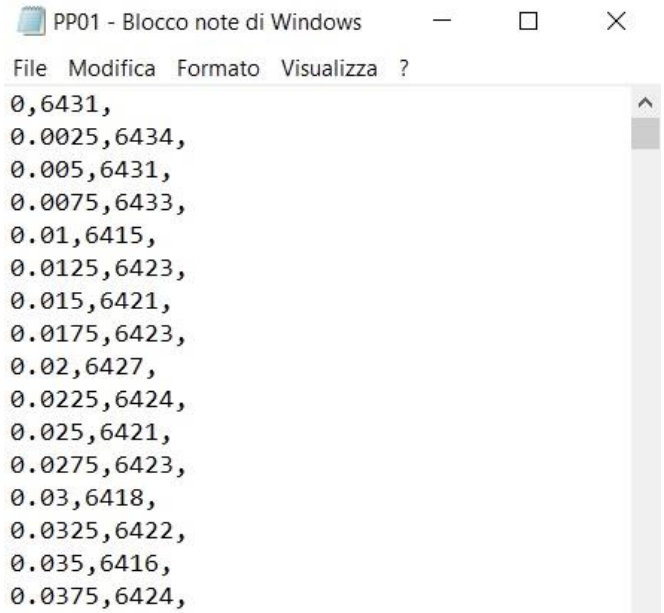

Fig. 7

- **AC**: dove la prima colonna è il tempo e le successive 3 colonne sono i segnali dell'accelerometro lungo x, y e z.

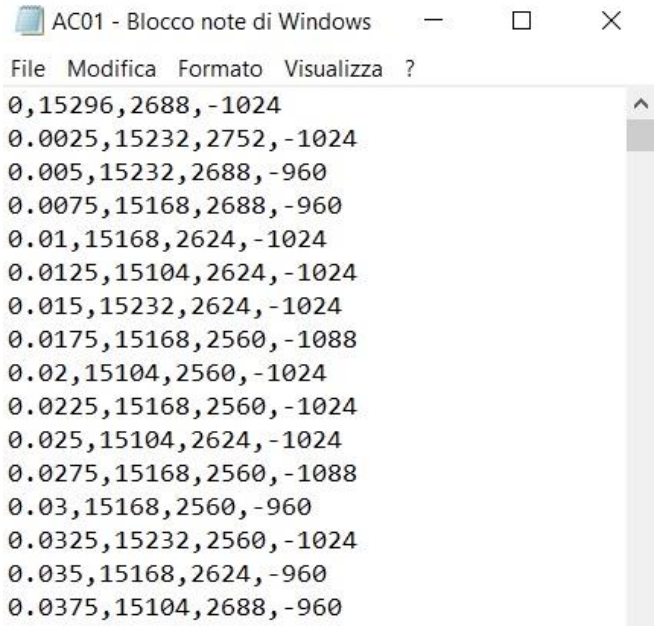

Fig. 8

- **HR**: dove la prima colonna è il tempo e la seconda è il vero HR

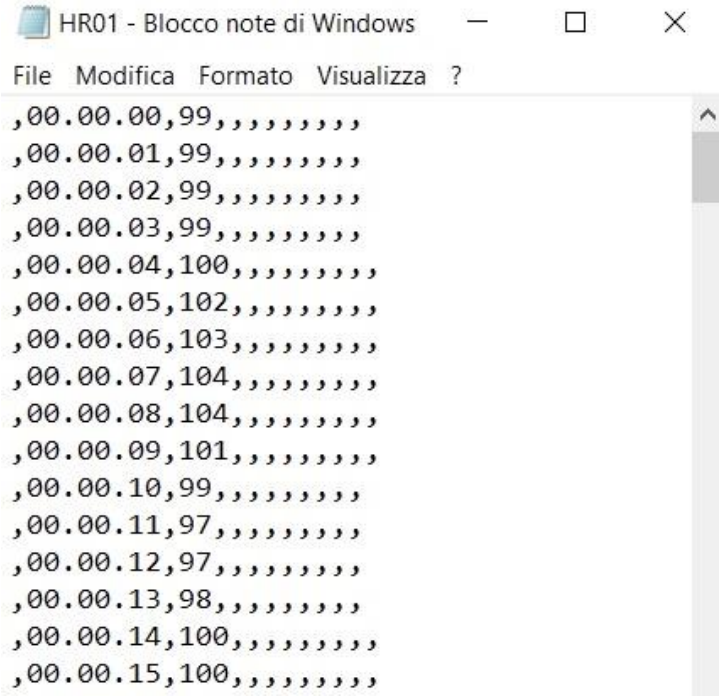

Fig. 9

Le figure mostrano, in piccola parte, i segnali PPG, AC e HR del soggetto 01 relativamente al gradino. Analoghi sono i file per quanto riguarda gli altri soggetti e anche per quanto riguarda lo squat.

3.3 Realizzazione

La prima cosa che è stata essenziale per svolgere il mio compito è stato studiare e provare l'algoritmo per capirne l'effettivo funzionamento e inoltre, con il fine di correggere eventuali bachi, una revisione è stata necessaria dato che il codice è stato utilizzato anche per altri scopi.
In secondo lungo è stata fondamentale anche la creazione di un semplice algoritmo genetico per testare con mano la loro implementazione su Matlab e anche per capire come era possibile andare a modificarne parametri quali il numero di generazioni, il numero della popolazione iniziale affinchè l'algoritmo genetico si comportasse come desiderato.

Dopo questo studio preliminare è stata necessaria una correzione del dataset che verrà giustificata durante la presentazione dello script principale utilizzato in Matlab: "extract_features" che dai 3 file prima citati, estrapola le caratteristiche principali del segnale utile da cui poi tracceremo l'HR.

```
global window;
global step;
global srate;
global l;

karma_setup
max_order = 20;

do_svd = true;
do_ideal_filter = false;

corpus.size       =  3;
corpus.path       = 'data/';
corpus.acfilename   = 'AC%02d.csv';
corpus.ppfilename   = 'PP%02d.csv';
corpus.hrfilename   = 'HR%02d.csv';
corpus.labels     = 'labels.txt';
cache.path        = 'cache/';
cache.filename     = 'DATA%02d_LFA.mat';

labels = load([ corpus.path corpus.labels ]);

global BPF;
global LPF;
len = length(BPF);
lag = (len - 1) / 2;
if length(LPF) ~= len
   error('Filter lengths mismatch!')
end

features = [];
```

Fino a qua sono state dichiarate le variabili e i percorsi relativi ai file sia di input sia di output, mentre da qui in poi vengono processati singolarmente i vari soggetti.

```
for subject = 1 : corpus.size

   fprintf('Processing file %02d ...', subject);
   filename_AC = [ corpus.path sprintf(corpus.acfilename, subject) ];
   filename_PP = [ corpus.path sprintf(corpus.ppfilename, subject) ];
   filename_HR   = [ corpus.path sprintf(corpus.hrfilename, subject) ];
   filename_out  = [ cache.path  sprintf(cache.filename, subject) ];
```

```
AC = load(filename_AC);
PP = load(filename_PP);
hr1=load(filename_HR);
sig = [AC'; PP'; PP'];
BPM0=zeros(56,2);
for i=1:2:112

      BPM0(i/2+1/2,1)=hr1(i,1);
      BPM0(i/2+1/2,2)=hr1(i,2);

end

hr = BPM0';
```

Qua sopra abbiamo caricato i file PP, AC e HR relativi al soggetto, si nota che la poiché "sig" contiene le trasposte di AC e PP, i due file .csv devono aver la stessa lunghezza ossia è necessario che arrivino allo stesso istante di tempo; in secondo luogo, per quanto riguarda il file dell'HR, nell' algoritmo viene finestrato ogni 2 s mentre le misurazioni avvenivano ogni secondo ma è bastato un ciclo for con passo due per risolvere ciò.
Qui di seguito avviene il filtraggio per ottenere le frequenze di interesse

```
hd = [ 0 diff(hr) ] / (step / srate);

start_time = tic;

history = [];
if do_ideal_filter
   sig_bpf = filter_window(sig, history, BPF);
   sig_lpf = filter_window(sig, history, LPF);
end
N = floor((size(sig, 2) - window) / step) + 1;
if N ~= length(hr)
   error('BPM length mismatch!');
end
F = cell(3, N);
L = cell(3, N);
A = zeros(2, N);
featurelet = zeros(N, 9);
for i = 1 : N
```

```
curSegment = (i-1)*step+1 : (i-1)*step+window;

if do_ideal_filter
   filtered_bpf = sig_bpf(end-4:end, curSegment);
   filtered_lpf = sig_lpf(end-4:end, curSegment);
else
   signal = sig(end-4:end, curSegment);
   filtered_bpf = filter_window(signal, history, BPF);
   filtered_lpf = filter_window(signal, history, LPF);
   history = [ history signal(:, 1:step) ];
   if size(history, 2) > lag
      history = history(:, end-(lag-1):end);
   end
end

if do_svd
   M = cell(1,5);
   for channel = 1 : size(filtered_bpf, 1)
      M{channel} = hankel(filtered_bpf(channel, 1:l), filtered_bpf(channel, l:end));
   end

   [ U{1}, lambda, ~ ] = svd([ M{1}          ], 'econ'); L{1,i} = diag(lambda);
   [ U{2}, lambda, ~ ] = svd([ M{2}          ], 'econ'); L{2,i} = diag(lambda);
   [ U{3}, lambda, ~ ] = svd([ M{3} M{4} M{5} ], 'econ'); L{3,i} = diag(lambda);
```

Nelle ultime righe qui sopra viene definita la matrice di Hankel ed effettuata la decomposizione ai valori singolari, mentre qui sotto, infine, viene calcolata la frequenza dominante di ogni componente, prima di andare a stampare quanti secondi ci ha impiegato tutto questo processo per il soggetto in questione e il salvataggio del file contenente le features ossia le frequenze dominanti dei vettori non appartenenti al sottospazio dei vettori di movimento, che saranno una prima stima grezza dell'HR prima del tracking.

```
   for channel = 1 : length(U)
      F{channel,i} = zeros(1, max_order);
      for p = 1 : max_order
         F{channel,i}(p) = frequency(U{channel}(:,p));
      end

   end
end
```

```
    A(:,i) = diag([srate 1]) * estimate_rotation_amount(filtered_lpf(3:5,:)')';

    label = ...
      labels(:,1) == subject & ...
      labels(:,4) * srate <  curSegment(1) & ...
      labels(:,5) * srate >= curSegment(end);
    label = labels(label, 3);
    if isempty(label)
      label = 0;
    else
      label = label(1);
    end
    featurelet(i, 1:3) = [ subject subject label ];
    featurelet(i,  4 ) = hr(i) >= 120 & hd(i) >= -0.1;
    featurelet(i, 5:6) = [ hr(i) hd(i) ] / 60;
    featurelet(i,  7 ) = F{3,i}(1);
    featurelet(i, 8:9) = A(:,i)';

  end

  process_duration = toc(start_time);
  fprintf(' took %.3f s\n', process_duration);

  features = [ features ; featurelet ];
  if do_svd
    save('-v7', filename_out, 'F', 'L', 'A');
  end

end

save('-v7', [cache.path 'features.mat'], 'features', 'srate', 'step', 'window');
```

3.4 Ottimizzazione

Lo script adoperato per l'ottimizzazione lo possiamo suddividere in 3 parti:

- La seguente matrice già presentata contenente gli 8 coefficienti, qua viene definita:

```
global min_max_starts;
min_max_starts0 = [ ...
   2      20     20      20    ;     ... % ppg_order (int)
   2      20     10      10    ;     ... % maf_order (int)
   0      10     2.0     2.0   ;     ... % maf_toler [BPM]
   0.5    0.99   0.8825  0.8825 ;   ... % alpha
   1      20     14      14    ;     ... % d_max    [BPM]
   0.01   0.99   0.8     0.8   ;     ... % k_range
   0.01   0.99   0.5     0.5   ;     ... % k_center
   0.01   0.99   0.12    0.12  ;     ... % k_fudge
];
```

- Qua definiamo le variabili e i vari percorsi dei file, "*searchables*" mi permette definire su quali degli 8 parametri voglio attuare l'ottimizzazione, in questo caso tutti e 8.
  Vengono, inoltre, definiti il *training set* e il *test set*.
  Con **Training Set** (o insieme di addestramento) indichiamo un insieme di dati che vengono usati per addestrare il sistema (la programmazione genetica è una particolare tecnica di apprendimento automatico che usa un algoritmo evolutivo per ottimizzare una popolazione di programmi di computer secondo un paesaggio adattivo determinato dall'abilità del programma di arrivare a un risultato computazionalmente valido), consiste in un vettore di input a cui è associata una risposta o una determinata classificazione; una volta effettuata la fase d apprendimento la correttezza dell'algoritmo viene verificata eseguendo lo stesso su un **Test Set** (o insieme di prova).
  Nel mio caso specifico i soggetti 2 e 3 formano il *training set* mentre il soggetto 1 è usato poi come *test set.*

```
cache.path = 'cache/';
load([cache.path 'models.mat'], 'test_folds');
numb_folds = length(test_folds);
subjects = 3;

searchables = { [1 2 3 4 5 6 7 8] };
ns = 1;
n0 = 1;

if n0 == 1
   progress = cell(ns,numb_folds);
   evolution = cell(ns,numb_folds);
else
   progress(n0:ns,:) = {[]};
end

   min_max_starts = min_max_starts0;
   if n0 > 1
      min_max_starts(:,3:4) = progress{n0-1,1};
   end
   test_subjects = [1];
   train_subjects = [2,3];
   k=1;
      params = [ min_max_starts(:,3)' min_max_starts(:,4)' ];
      e0_test = track(params, test_subjects);
      do_optimization(searchables{k}, train_subjects);
      params = [ min_max_starts(:,3)' min_max_starts(:,4)' ];
      e1_test = track(params, test_subjects);
      fprintf('Optimization %d progressed test set from %f to %f.\n', k, e0_test,
e1_test);
      progress{k,1} = min_max_starts(:,3:4);
      evolution{k,1} = trajectory;
```

- Ultima, ma non per importanza, è la definizione della funzione "*do_optimization*", utilizzata prima sul *training set.* È la funzione che richiama l'algoritmo genetico vero. "*optimoptions*" mi permette di andare a impostare le caratteristiche dell'algoritmo genetico, in particolare il numero di generazioni massimo a 10, che nel concreto blocca l'algoritmo alla 10 generazione.

```
function do_optimization(searchables, subjects)
global min_max_starts;
n = size(min_max_starts, 1);
fixed_map = ones(1, n);
fixed_map(searchables) = 0;
fixed_map = [ fixed_map fixed_map ];
scatter = eye(n);
scatter = scatter(searchables,:);
scatter = [ scatter, scatter];
initial = [ min_max_starts(:,3)' min_max_starts(:,3)' ];
mms = min_max_starts(searchables, :);
fx = @(x) track(initial .* fixed_map + x * scatter, subjects);
y0 = track(initial, subjects);
r = length(searchables);
intcont = find(ismember(searchables, [1]));
intcont = [ intcont ];

opts = optimoptions('ga', ...
   'OutputFcn', @gaoutfun, ...
   'MaxGenerations', 9 ...
);
[x,y] = ga( ...
   fx, r, [], [], [], [], ...
   [ mms(:,1)' ], ...
   [ mms(:,2)' ], ...
   [], intcont, opts ...
);
min_max_starts(searchables,3) = x(1:r)';
min_max_starts(searchables,4) = x(1:r)';
fprintf('Optimization progressed train set from %f to %f.\n', y0, y);
end
```

Arrivato al numero di "MaxGenerations" impostato, l'algoritmo termina e troveremo i valori degli 8 coefficienti all'interno della variabile "params" accessibile dal *Workspace.*

4. CONCLUSIONI

4.1 Risultati ottenuti

Con il fine di ottenere un errore accettabile è stato necessario fare delle prove andando a modificare il numero di generazioni a cui doveva arrivare l'algoritmo genetico (il numero di individui è stato lasciato di default ossia 100).
In questo caso di prova il numero massimo di generazioni era **4**:

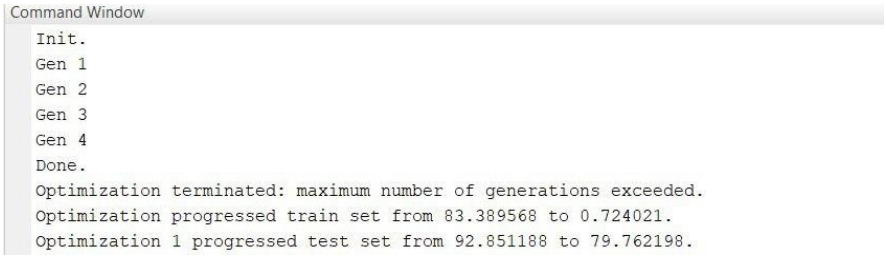

FIg. 10

Settandolo invece a **9** ci troviamo nella situazione desiderata:

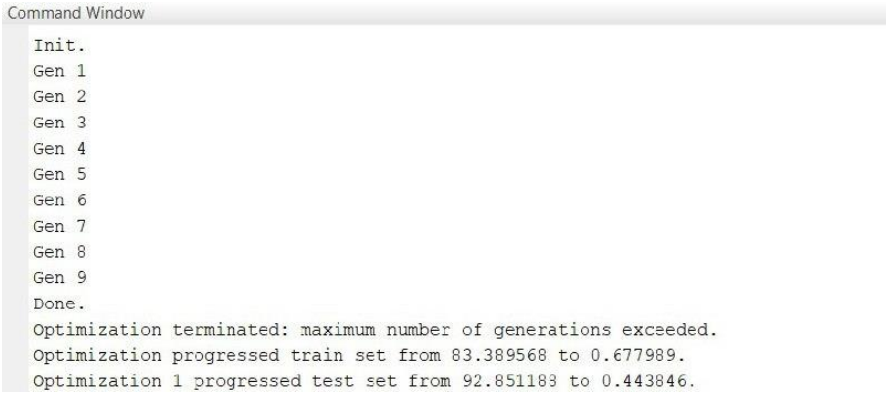

Fig. 11

PARAMETRI OTTIMIZZATI PER IL GRADINO:

- *ppg_order:* **19**
- *maf_order:* **3**
- *maf_toler:* **0.9651**
- *alpha:* **0.9891**
- *d_max:* **14.5156**
- *k_range:* **0.4030**
- *k_center:* **0.3028**
- *k_fudge:* **0.1500**

PARAMETRI OTTIMIZZATI PER LO SQUAT:

- *ppg_order:* **15**
- *maf_order:* **2**
- *maf_toler:* **0.9720**
- *alpha:* **0.8728**
- *d_max:* **11.7660**
- *k_range:* **0.9236**
- *k_center:* **0.9173**
- *k_fudge:* **0.0174**

Dopo aver ottenuto questi risultati bisogna comunque fare delle considerazioni riguardo a quello che abbiamo ottenuto.

Essendo il dataset formato da soli 3 soggetti, l'ottimizzazione è avvenuta, ma si potrebbe espandere il numero di soggetti o il numero di misurazioni per ogni soggetto relativamente agli squat e al gradino in modo tale da smussare questi valori ulteriormente.

Perché si dovrebbero pulire ulteriormente questi valori?

Perché essendo limitati a 3 soggetti, si potrebbe andare in contro al fenomeno dell'***Overfitting*** (adattamento eccessivo o sovradattamento) che si verifica quando un modello molto complesso si adatta troppo ai dati osservati perché ha un numero eccessivo di parametri rispetto al numero di osservazioni.
(Anche per questo il numero di generazioni è stato limitato a 9)

Una soluzione che si può attuare è la ***Cross-vaidation*** (convalida incrociata) che è una tecnica statistica utilizzabile in presenza di una buona numerosità del *training set*.

L'algoritmo principale con i 12 soggetti presentava questa implementazione; Il dataset viene diviso in 4 cartelle e in ogni cartella 3 soggetti sono scelti come test set e i rimanenti 9 soggetti usati come training cosicché ogni soggetto venga usato per testare in un' unica cartella.
L'ottimizzazione dei parametri è poi fatta indipendentemente su ogni cartella usando il sottoset di soggetti dedicati al training, poi usati per valutare le performance della stessa cartella di test set.

In particolare, la ***k-fold cross-validation*** consiste nella suddivisone del dataset totale in k parti di uguale numerosità e, ad ogni passo, la k-esima parte del dataset viene ad essere il *validation dataset*, mentre la restante parte costituisce il *training set*. Così, per ognuna

delle k parti si allena il modello, evitando quindi problemi di *overfitting*.

In altre parole, si suddivide il campione osservato in gruppi di egual numerosità, si esclude iterativamente un gruppo alla volta e lo si cerca di predire con i gruppi non esclusi. Ciò al fine di verificare la bontà del modello di predizione utilizzato.

Qui di seguito: la curva blu mostra l'andamento dell'errore nel classificare i dati di *training*, mentre la curva rossa mostra l'errore nel classificare i dati di *test* o *validazione*. Una situazione in cui il secondo aumenta mentre il primo diminuisce è indice della possibile presenza di un caso di *overfitting*.

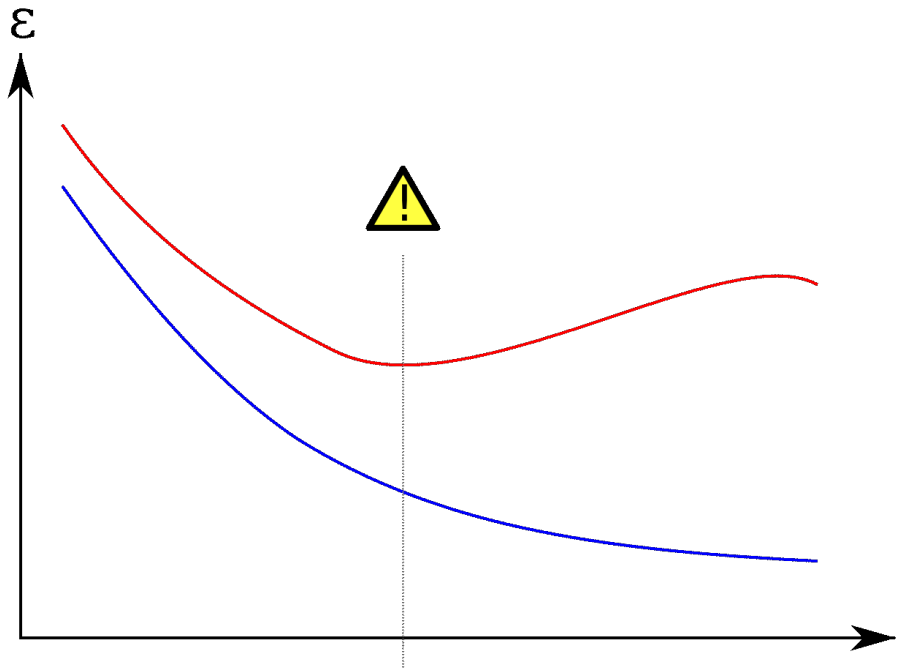

Fig. 12

4.2 Sviluppi futuri

Ulteriori sviluppi sono l'ottimizzazione di questi parametri relativamente ad altre attività, anche poco impegnative, della vita di tutti i giorni che, nel concreto, si potrebbero utilizzare per monitore pazienti o persone di una certa età.

Si potrebbero ottimizzare anche relativamente alle attività fisiche che si svolgono in palestra o durante sport specifici.

In fondo l'obbiettivo unico di tutto l'approccio è stato un calo di precisione nel tracking del HR a favore di una bassa complessità computazionale, ma è intuitivo il fatto che si riesce a sopperire al calo di precisione nel momento in cui i coefficienti sono stati ottimizzati per un vasto numero di attività.

Elenco figure:

Bibliografia

- "Reduced Complexity Algorithm for Heart Rate Monitoring from PPG Signals using Automatic Activity Intensity Classifier" del DII-Dipartimento di Ingegneria dell'Informazione, Università Politecnica delle Marche: Giorgio Biagetti, Paolo Crippa, Laura Falaschetti, Simone Orcioni, Claudio Turchetti.

Sitografia

- https://it.mathworks.com
- https://it.wikipedia.org

Ringraziamenti

Ringrazio il Prof. Paolo Crippa per la disponibilità dedicata per lo sviluppo di questa tesi e per aver dato la possibilità di affrontare temi completamente nuovi.
Ringrazio il Prof. Giorgio Biagetti per avermi concesso gli strumenti, le conoscenze oltre che il suo tempo e la sua pazienza per lo sviluppo di questa tesi vista la sua disponibilità in ogni circostanza.