Simone Roznowicz 1216375

PROGETTO VAD

E’ stato realizzato un algoritmo adattativo per la rilevazione della voce umana: a seguito della pacchettizzazione dei campioni, esso è in grado di rilevare la presenza di rumore (prevalentemente stazionario) e ritornare un vettore contenente, per ogni pacchetto, un 1 (rilevata voce) o uno 0 (rilevato rumore).
Pertanto l’algoritmo permette il riconoscimento vocale, con buoni risultati, su un generico file audio che presenta rumore (stazionario soprattutto).

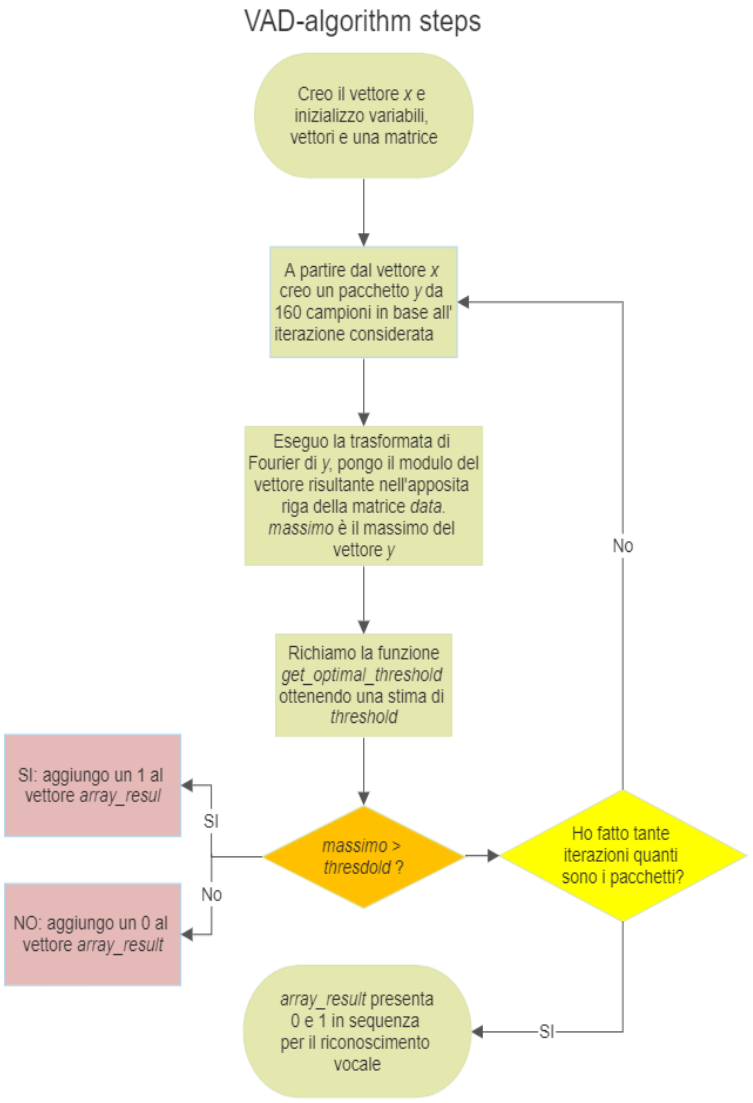
ADATTAMENTO e CRESCENTE AFFIDABILITÀ
L’algoritmo e’ adattativo perché applica ragionamenti statistici per la distinzione, pacchetto dopo pacchetto, del rumore dalla voce. Più precisamente il cuore dell’algoritmo si fonda sulla funzione get_optimal_threshold che fornisce, ad ogni iterazione, un threshold, sempre più preciso, utilizzato per stimare la bontà di ogni pacchetto.
L'affidabilità dell’algoritmo cresce quindi col tempo, in quanto esso può basare la decisione in merito al pacchetto n sulle informazioni riguardanti i pacchetti da 1 a n. Pertanto, si possono verificare occasionali errori iniziali che diventano statisticamente sempre più improbabili fornendo più pacchetti da analizzare.
Di base l’algoritmo sfrutta la deviazione standard fra i campioni all’interno di ogni pacchetto in modo da rilevare il rumore gaussiano, piuttosto costante, che caratterizza gli audio. Inoltre esso ricorre al massimo di ogni pacchetto (sempre nell’analisi in frequenza) per stimare la soglia (threshold aggiornato a ogni iterazione) a cui arriva il rumore ad ogni audio. In tal modo, qualora il massimo di un pacchetto fosse maggiore di tale soglia, con ogni probabilità si sarebbe di fronte a un campione rilevante (voce umana) e verrebbe scritto un 1 nel vettore *array_result.*

FUNZIONE get_optimal_threshold: GENERALITA’ E STRUTTURA
parametri di input: *data, threshold, coeff, max_estimate, max_estimate_list, sigma_estimate, numero_pacchetti_incontrati*
parametri di output: [*threshold, max_estimate, max_estimate_list, sigma_estimate, numero_pacchetti_incontrati*]
(Threshold valore di ritorno principale, è necessario ritornare anche gli altri parametri per permetterne l’aggiornamento ad ogni iterazione).

E’ fondamentale evidenziare che la funzione è realizzata in modo da consentire la voice detection in tempo reale: in altre parole a ogni iterazione del ciclo while viene restituito uno 0 o un 1 in output. Questo approccio generale permette di ridurre al minimo il tempo impiegato per prendere una decisione per ogni pacchetto. Pertanto la funzione non è condizionata dai pacchetti non ancora considerati (proprio per cercare di emulare al meglio ciò può avvenire in tempo reale). I rari pacchetti non correttamente classificati all’inizio di un file audio diventano trascurabili quanti più campioni vengono considerati.

La funzione get_optimal_threshold effettua considerazioni diverse in base al *numero_pacchetti_incontrati* a una data iterazione: vengono gestiti separatamente i casi

iniziali in cui *numero_pacchetti_incontrati* <= 2 (i parametri sono fissati a delle soglie arbitrarie che vengono poi costantemente aggiornate).
A ogni nuova osservazione aggiorno *sigma_estimate* (tiene conto della STD dei pacchetti incontrati) se il residuo tra *this_sigma* (STD del pacchetto in questione) e *sigma_estimate* è sotto una percentuale scelta di *sigma_estimate* (vedere il codice). Quando viene aggiornato *sigma_estimate*, viene fatto similmente per il massimo che fornisce una misura più variabile (outlier); Di tale imprecisione viene tenuto conto alla fine nella restituzione di un threshold pari a *max_estimate* sommato a 5*std (max_estimate_list).
Di fatto, grazie a *sigma_estimate* e *max_estimate*, viene individuato il rumore in un intervallo di valori che dipende dalle oscillazioni di entrambi.



ESEMPIO PER **inputaudio2.data**
Sono riportati in seguito i due grafici relativi al modulo del vettore *x* (che contiene tutti i campioni dei vari input audio) originale (in rosso) e filtrato (in blu).
Quest'ultimo è ottenuto azzerando tutti i campioni appartenenti a pacchetti reputati rumore dall'algoritmo VAD (sfruttando quindi *array_result*).
Si può notare anche visivamente come il rumore di fondo sia stato rimosso, lasciando inalterato il contributo della voce al segnale originario. Dal momento in cui *length(x)* non è multiplo di *packet_length*, considero *num_packets = floor(length(x)/packet_length);* di fatto non considero l'ultimo pacchetto parziale, sia per non aggiungere campioni nulli allungando il vettore sia perchè la perdita è assolutamente trascurabile (<20ms).
Nell'esempio vengono considerati 294 pacchetti invece che 294.4.