POLITECNICO DI MILANO

FACOLTÀ DI INGEGNERIA INDUSTRIALE E DELL'INFORMAZIONE

Corso di Laurea Magistrale in Ingegneria Informatica

Sviluppo e implementazione di un classificatore per dispositivo indossabile basato su segnale sEMG per movimenti di arto superiore

Relatore: Prof.ssa Giuseppina GINI
Correlatore: PhD Paolo BELLUCO

Tesi di Laurea Magistrale di:
Lisa MAZZON Matr. 800536
Simone PONTIGGIA Matr. 799357

Anno Accademico 2014/2015

Alle nostre famiglie

Sommario

Protesi ed esoscheletri hanno il compito di sostituire una parte del corpo o di incrementarne le abilità motorie; per controllare simili dispositivi le intenzioni di moto possono essere ottenute da un sistema mioelettrico. Tale sistema utilizza spesso un metodo di *machine learning* chiamato *Pattern Recognition*, che riconosce dei modelli all'interno del segnale in ingresso in modo tale da poterli poi individuare e classificare su un segnale nuovo. In questo studio di tesi si vuole indagare fino alla fase di classificazione del segnale e quindi poter discriminare le diverse classi di movimento. Tale studio verrà applicato per distinguere nove movimenti della spalla, perché questa articolazione è stata poco indagata in letteratura, visto che comprende muscoli poco superficiali e presenta alto *cross-talking*. Verrà prima sviluppato e in seguito implementato un metodo di classificazione per discriminare le classi di movimento, con lo scopo di essere impiegato all'interno di un dispositivo indossabile (*Intel Edison*) mediante l'utilizzo di un segnale EMG di superficie. I test sono stati effettuati prima su rilevazioni sEMG ad otto canali, acquisite per mezzo di un sistema professionale, e successivamente mediante una scheda di acquisizione a tre canali chiamata *Eracle*, ideata e sviluppata all'interno del Politecnico di Milano dal PhD Paolo Belluco. I risultati finali mostrano come configurazione ideale una segmentazione di lunghezza *500 ms* e offset di *62 ms*, l'utilizzo del set di caratteristiche di *Hudgins* (MAV,SSC,WL,ZC) e un nuovo approccio in ambito sEMG per il classificatore, basato sulle reti neurali feed forward e definito come *classificatore a due fasi*, che aumenta le prestazioni a real-time. Inoltre viene mostrata la possibilità di ridurre i canali da otto a tre ottenendo un'accuratezza fino al 97.2%. Le prestazione temporali del percorso scelto sono di *17 ms* per segmento, ben al di sotto del requisito real-time (*300 ms*). Sono stati effettuati inoltre dei test con un dataset della mano, al fine di convalidare la generalità della classificazione ideata. Questo studio è stato effettuato all'interno del Dipartimento di Elettronica, Informazione e Bioingegneria presso il Politecnico di Milano.

Abstract

A prosthesis or an exoskeletal application, is intended to replace a body part or work as an aid in motion, to work those devices need a myoeletrical sistem. These systems are based on a *machine learning* method called *Pattern recognition*, that identifies patterns in the myoeletric acquisition for a later classification on a new incoming signal. This thesis is focused on the classification phase, where is possible to distinquish each class of motion. The study of the classification phase aims to identify nine different movements of the shoulder because this body part hasn't been completely treated in literature, considering that shoulder muscles are less superficial and the signal collected is noisy (*due to cross-talking*). It will be developed and afterwards implemented a methods for motion classification with the intent of integrate it on a wearable device (*Intel Edison*). Several tests has been done on sEMG data, first acquired using an eight channel professional acquisition system and after using a three channel acquisition board called *Eracle*, designed and developed at the Politecnico di Milano by PhD Paolo Belluco. The final results show that the ideal configuration is made of a segmentation with a window of lenght *500 ms* and offset of *62 ms*, extracting *Hudgins* feature set (MAC, SSC, WL, ZC) and a novel sEMG classification approach, based on feed forward neural network, defined as *multiphase classifyer*, that will enhance real time classificazion accuracy. It has been further investigated the possibility of a reduction of the acquisition channels from eight to three remaining on a 97.2% classification accuracy. The performance of the proposed classification path, has been 17ms per segment analysed, far less than the given real time constraint of *300 ms*. Have been made tests with a hand dataset with the aim of document the capability of the proposed system to perform well also on different applications. This study as been realized at the Department of Electronics , Information and Bioengineering in Politecnico di Milano.

Indice

Elenco delle figure

Elenco delle tabelle

1. Introduzione

> Theory is when you know everything but nothing works. Practice is everything works but no one knows why. Here, theory and practice are combined: nothing works and no one knows why
>
> Albert Einstein

Nell'ambito medico di assistenza agli amputati le protesi attive ricoprono una fetta importante di mercato, in quanto consentono di rimpiazzare l'arto perso con un suo equivalente funzionale. Negli anni si è sentito il bisogno di protesi sempre più evolute che riescano a effettuare un numero sempre maggiore di movimenti, diventando sempre più simili all'arto originale. Mentre sono da tempo disponibili sul mercato le protesi attive di mano della Ottobock, che permettono movimenti di apertura e chiusura, più recentemente sono state introdotte protesi di braccio per amputazioni a livello trans-omerale e le prime protesi di spalla che consentono di attuare due gradi di libertà (ad esempio la protesi *INAIL* nel 2008). Ma l'attenzione non è rivolta solamente alle protesi attive, esiste anche un notevole interesse per quanto riguarda il settore degli esoscheletri, che sono dispositivi di assistenza motoria. Un esoscheletro viene utilizzato per sopperire o migliorare le capacità motorie di un soggetto: nel primo caso può essere utilizzato durante la riabilitazione per guidare gradualmente il paziente verso il recupero delle capacità motorie, mentre nel secondo caso può essere utilizzato in ambito

militare o in situazioni di soccorso, per potenziare le abilità del soldato o del volontario addetto all'operazione.

Entrambi questi dispositivi hanno bisogno di ricevere un segnale di controllo da parte dell'utente, ed una strategia particolarmente utilizzata in letteratura è quella basata sull'analisi del segnale elettromiografico. Questo segnale viene prodotto spontaneamente dai muscoli del corpo umano durante la sollecitazione e può essere misurato con degli elettrodi che rilevano la differenza di potenziale elettrico tra due punti posti generalmente sulla direzione di massima variazione, ossia parallelamente alla fibra muscolare. Il segnale EMG (ElettroMioGrafico) viene prodotto dalla contrazione indotta o involontaria dei muscoli scheletrici e più precisamente dalle correnti ioniche che fluiscono lungo le membrane delle fibre muscolari. La variazione di potenziale all'interno del muscolo avviene per mezzo delle unità di movimento, chiamate unità motorie, le quali vengono reclutate per compiere un movimento: maggiore è il numero di unità reclutate (o la loro frequenza di reclutamento), maggiore sarà la forza esercitata dal muscolo.

Per l'analisi del segnale elettromiografico, essendo un segnale stocastico, non sono disponibili modelli puntuali che descrivano le variazioni del potenziale nel tempo: per questo motivo una strategia comune in letteratura è di utilizzare degli strumenti che apprendano le dinamiche del segnale e possano fornire informazioni utili ad identificare i movimenti o l'intenzione di movimento. Questa strategia di identificazione è vincente dato che si basa solo sul segnale EMG, il quale viene prodotto spontaneamente dal corpo umano. Recentemente per migliorare la classificazione dei movimenti, si utilizza una tecnica denominata *Elaborazione Multicanale*, la quale utilizza l'acquisizione di più segnali elettromiografici provenienti da diversi muscoli con il risultato di ottenere un'accuratezza eccellente anche su molti movimenti. Quest'ultima tecnica, come verrà ripetuto in seguito, è per ora rilegata all'ambito diagnostico, non essendo disponibile ad oggi un sistema di classificazione real-time e multicanale per i movimenti della spalla (parte del corpo maggiormente analizzata in questo studio di tesi).

Le strategie basate su sEMG vengono utilizzate correntemente in quasi tutte le protesi attive sul mercato e in buona parte negli esoscheletri disponibili, come l'esoscheletro *HAL* della giapponese Cyberdyne; questa tecnica è così diffusa perché non è invasiva per il paziente, utilizzando l'EMG prelevato in superficie, consente di rilevare informazioni anche dai muscoli residui a seguito di una amputazione e costituisce il modo più naturale per il paziente di utilizzare la protesi o l'esoscheletro. Ma anche se questo approccio è noto da anni non esiste ancora una corretta ingegnerizzazione del problema: molte delle protesi commerciali hanno una limitata capacità di movimento, dovuta a pochi giunti controllabili, acquisiscono un segnale elettromiografico

da pochi elettrodi e di conseguenza possono riconoscere pochi movimenti. Queste limitazioni, se non sono così drammatiche per una protesi di mano, lo sono invece per una protesi o un esoscheletro relativo alla spalla, dato l'alto numero di movimenti che questa parte del corpo può compiere. Occorre quindi sviluppare delle soluzioni che consentano di aumentare il numero di informazione estraibile dal segnale muscolare per poter riconoscere sempre più movimenti.

Come già si intuisce, non solo è importante avere un buon dispositivo fisico, ma soprattutto il software è di cruciale importanza. Se è vero che una buona protesi può consentire virtualmente di effettuare movimenti fluidi come un vero braccio umano, o di amplificare forza e precisione di un soldato o un soccorritore, è il software, responsabile dell'identificazione dei movimenti, a consentire di controllare l'hardware in modo ottimale. Chiarito il ruolo fondamentale che il software di classificazione svolge nella costruzione di una protesi elettromiografica o di un esoscheletro, è doveroso notare la carenza di classificatori capaci di identificare più di due o tre classi di movimento durante una sessione online. Se per la costruzione di una protesi di mano questa limitazione può essere tuttavia accettabile, compromette decisamente lo sviluppo di protesi di arto superiore completo, o la costruzione di un esoscheletro di assistenza ai movimenti della spalla.

Considerando che la maggior parte delle amputazioni all'arto superiore, avvengono a livello trans-omerale e trans-radiale, risulta chiaro come mai il problema dei sistemi mioelettrici per disarticolazione alla spalla non sia ancora stato trattato completamente in letteratura. Inoltre, da un punto di vista più tecnico, l'analisi del segnale sEMG per la spalla presenta delle criticità dovute alla sua conformazione complessa: i muscoli della spalla sono meno superficiali e quindi forniscono un segnale meno forte agli elettrodi, e la particolare localizzazione dei muscoli produce un elevato *cross-talking*.

Nonostante queste difficoltà c'è la necessità di affrontare il problema, poiché maggiore è il livello di amputazione o deficit motorio nell'arto superiore, più il paziente ha necessità di una protesi attiva o un esoscheletro, non avendo alcuna parte di arto rimasta utilizzabile.

Le motivazioni illustrate sopra hanno spinto ad intraprendere questo lavoro di tesi, focalizzandosi sulla progettazione di un classificatore elettromiografico capace di acquisire il segnale sEMG e identificare i movimenti a livello dell'articolazione della spalla. Attualmente esistono un certo numero di studi riguardo all'elaborazione del segnale elettromiografico per disarticolazione alla spalla, ma tutti affrontano il problema da un punto di vista diagnostico o di analisi. Per questo si è deciso di concentrarsi su progettazione e sviluppo di un classificatore real-time, adatto ad essere impiegato in una futura applicazione protesica o su un esoscheletro.

Gli obiettivi principali di questo studio di tesi comprendono: la progettazione di un classificatore che consenta di identificare correttamente nove diversi movimenti, implementato su un dispositivo indossabile dalle dimensioni ridotte, che riesca a eseguire il ciclo di classificazione con una frequenza abbastanza elevata da consentire il controllo della protesi in un tempo minore di $300ms$, dall'acquisizione del segnale fino alla attuazione del dispositivo fisico. Data l'importanza del sistema di classificazione all'interno di una protesi o un esoscheletro e per fornire un risultato applicabile su più fronti, si è deciso di concentrare lo studio su questo componente, senza prendere in considerazione la progettazione del controllore o l'analisi delle strategie di controllo.

Rispetto ai lavori presenti in letteratura il dispositivo proposto in questa tesi consente di estendere le capacità di classificazione del segnale sEMG al mondo dei dispositivi indossabili, ampliando il numero di movimenti riconoscibili fino a nove, e dimostrando di poter operare con un numero minore di canali rispetto alle controparti per uso diagnostico senza un degrado significativo della capacità di classificazione. Questi aspetti consentono di affermare che le tecnologie disponibili sono mature per lo sviluppo di protesi o esoscheletri elettromiografici multicanale di una certa complessità per arto superiore.

Partendo da un lavoro di tesi redatta nel 2013 presso il Politecnico di Milano da Rivela e Scannella [1], si è deciso di svolgere una trattazione più approfondita, valutando nuove progettazioni e metodologie per la risoluzione del problema e applicando i risultati ottenuti su un sistema indossabile (*Intel Edison*).

Dopo aver definito uno stato dell'arte in fatto di Pattern Recognition per la classificazione, sono stati svolti degli studi teorici e pratici sull'analisi del segnale sEMG, definendo la segmentazione, la scelta dell'insieme di caratteristiche da utilizzare e i metodi di pre-processing che più si adattavano a questa trattazione. É stata fatta una valutazione del contenuto informativo per ogni canale di un dataset acquisito nello studio da cui si è partiti [1], al fine di avere maggiori informazioni sull'acquisizione del segnale sEMG dei muscoli della spalla. É stata scelta l'architettura del sistema, effettuando una differenziazione della fase di training da quella di testing, implementando la prima con lo scopo di utilizzarla su un laptop o un PC desktop e la seconda su un dispositivo indossabile (Intel Edison), scelto con criterio per essere economico, leggero e con bassi consumi.
Per la comunicazione tra le due macro-componenti del sistema si è optato per un formato di facile utilizzo, che consentirà espansioni successive del sistema di classificazione.

Sono stati valutati diversi tipi di classificatori, lineari e non lineari, ideando anche un nuovo approccio in ambito di segnale sEMG, composto da una classificazione a rete neurale a due fasi, che in seguito verrà descritto più nel dettaglio.
Infine si è scelto di effettuare una nuova acquisizione del segnale utilizzando un dispositivo meno professionale (Eracle), al fine di testare quanto scoperto in situazioni meno ottimali e in presenza di rumore.

La trattazione si articola nei successivi sette capitoli.

Nel Capitolo 2 vengono descritte le diverse fasi della Pattern Recognition, per avere una visione generale di ciò che viene utilizzato oggi in letteratura.

Al Capitolo 3 vengono esposti gli obiettivi dello studio di tesi e le scelte progettuali intraprese, descrivendo le architetture del sistema di classificazione e l'hardware utilizzato.

Nel Capitolo 4 è presente una trattazione dei dataset usati e una loro valutazione dal punto di vista del numero di canali necessari per la loro acquisizione.

Il Capitolo 5 invece descrive le scelte progettuali per lo sviluppo del sistema, valutando il rendimento di quelle selezionate, e la fase di training svolta con lo strumento Matlab.

Nel Capitolo 6 viene esposta l'implementazione sul dispositivo indossabile *Intel Edison*, valutando le performance dal punto di vista dell'accuratezza e delle tempistiche.

Al Capitolo 7 vengono descritti i risultati ottenuti da questo studio, riassumendo quanto trattato ai capitoli precedenti ed esponendo nuovi test per mostrare gli obiettivi raggiunti.

Nel Capitolo 8 vengono esposte le considerazioni finali, valutando nell'insieme il percorso svolto e proponendo nuove applicazione future.

In Appendice A sono presenti i riferimenti ai repository in cui si trova il codice utilizzato.

2. Pattern Recognition per il segnale EMG

> Any sufficiently advanced technology is indistinguishable from magic
>
> A. Clark

Watanabe [2] definisce un pattern *"as opposite of a chaos; it is an entity, vaguely defined, that could be given a name"*. Il compito del pattern matching è di prendere un insieme di dati scorrelati e statisticamente piatti e far emergere le loro individualità, dando loro un nome e classificandoli così in categorie (o classi).

Un sistema di classificazione basato sulla Pattern Recognition si articola nelle seguenti fasi:

- acquisizione del segnale sEMG e preprocessing
- elaborazione dei dati (segmentazione del segnale, rappresentazione delle caratteristiche)
- classificazione del movimento mediante il riconoscimento della classe

I problemi di questo tipo richiedono molte scelte progettuali, tra cui la tipologia di sensori, le tecniche di preprocessing, lo schema di rappresentazione e il modello di decisione da utilizzare.

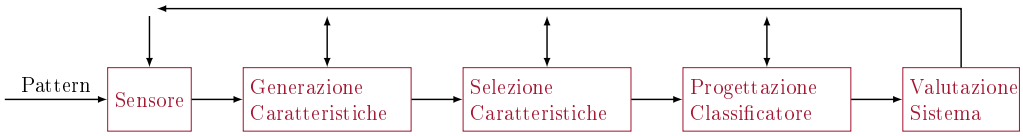


Figura 2.1: Questo schema illustra le fasi che devono essere affrontate nella costruzione di un classificatore per il pattern matching

In letteratura viene sempre evidenziata l'importanza di ben definire il problema di riconoscimento. Un problema ben definito porterà ad una compatta rappresentazione dei pattern e ad una semplice strategia di decisione, ma richiede anche che la sua rappresentazione abbia piccole variazioni intra-classe e grandi variazioni extra-classe. Nei sistemi di pattern matching si sceglie spesso di acquisire un dataset (*Set di Training*) ed utilizzarlo come insieme di esempio per costruire la strategia di decisione [3].

Nelle sezioni seguenti verranno esplorate le varie fasi della Pattern Recognition per la classificazione, valutando i metodi presenti oggi in letteratura. Tali fasi analizzano il segnale sEMG fino ad arrivare a un riconoscimento della classe che corrisponde al determinato movimento. Nel presente lavoro non verrà esplorata la fase di controllo, poiché l'idea è quella di poter impiegare il sistema analizzato sia in protesi che in esoscheletri per il supporto delle attività quotidiane.

2.1 Acquisizione del segnale EMG

Il segnale EMG, da utilizzare ad esempio per una protesi mioelettrica, può essere prelevato in superficie (sEMG) o mediante elettrodi impiantati direttamente all'interno del tessuto muscolare.
Attraverso il metodo intramuscolare si avrà un segnale con minore cross-talking, con la possibilità di raggiungere muscoli più in profondità e avere una maggiore selettività, riuscendo così ad effettuare una classificazione più accurata.
Il problema principale per questo tipo di acquisizione è la sua invasività, che comporta possibili svantaggi quali infezioni, manutenzione, alto costo, poca tollerabilità da parte del paziente: è per questo che oggi si predilige l'utilizzo degli elettrodi di superficie. Inoltre diversi studi [4] [5] dimostrano che, scegliendo le giuste caratteristiche e un classificatore appropriato, l'acquisizione del segnale (se svolta correttamente) è la fase che incide meno sull'errore di classificazione.

Per il prelievo di sEMG vengono utilizzati degli elettrodi applicati all'interno dell'invasatura della protesi, costruita rispettando l'anatomia del paziente. Difatti le protesi mioelettriche maggiormente usate sono multicanale, al fine di ottenere maggiori informazioni per la classificazione. Diversi studi [6] dimostrano come l'aumento di canali provochi una riduzione dell'errore di classificazione.
L'introduzione di più canali, però, provoca un aumento del peso, della complessità e del costo della protesi, per cui oggi è importante riuscire a trovare un compromesso tra la diminuzione del numero di canali e l'informazione ricevuta dai segnali.
Nella tesi di Rivela e Scannella [1] i canali utilizzati sono otto (per il riconoscimento di nove classi), ma in casi reali il numero di canali si riduce (ad esempio al massimo quattro per la protesi INAIL [7]), diminuendo però il range di movimenti classificati.

Purtroppo il segnale sEMG presenta degli svantaggi, in particolar modo per la protesi alla spalla.
Questo tipo di protesi deve avvalersi di muscoli superficiali poco selettivi, che presentano elevato cross-talking, e anche un leggero spostamento dell'elettrodo o una pelle non del tutto pulita possono provocare gravi distorsioni del segnale, con successivo errore di classificazione.
Negli ultimi anni si è sviluppato un nuovo metodo chirurgico chiamato *Targeted Muscle Reinnervation* (TMR) [8], con lo scopo di aiutare l'acquisizione del segnale EMG per la protesi mioelettrica.
Questa procedura rende nuovamente utilizzabili i nervi che controllavano braccio e mano, riassegnandoli ad alcuni tratti poco usati dei muscoli pettorali. Viene effettuata una denervazione della parte del muscolo target e successivamente viene reinnervato con le fibre appartenenti un tempo ai nervi che si occupavano del controllo del braccio.
Questo consente di avere un segnale migliore e di ricevere anche un feedback sensoriale.
La TMR, anche se in continua espansione, è una tecnica non sempre attuabile perché dipende dalla fisionomia del paziente e dalla scelta dello stesso, poiché rimane pur sempre un'operazione chirurgica di esito incerto, con i rischi che ne conseguono.

In generale, il segnale acquisito dai muscoli di un paziente, con o senza TMR, dovrà essere filtrato con un passa-banda, al fine di limitare il rumore [1].
Innanzitutto viene applicato un filtro passa-basso per attenuare le alte frequenze indesiderate ed evitare il fenomeno di *aliasing*. Tale filtro viene definito rispettando il teorema di campionamento di Nyquist, il quale afferma

che la frequenza di taglio deve essere:

$$f_{cut-off} \leq \frac{f_{sampling}}{2} \tag{2.1}$$

dove $f_{sampling}$ corrisponde alla frequenza di campionamento del segnale, che negli studi EMG solitamente è di 1000 Hz [9] [10] [11]). Una così alta frequenza però determina una maggiore complessità computazionale (e quindi maggior tempo per l'elaborazione), per cui studi più recenti [12] cercano di dimezzare tale frequenza (500 Hz), ottenendo dei risultati con un'accuratezza di poco inferiore rispetto ai 1000 Hz solitamente usati (in media si riduce del 2%).
Oltre a questo filtro, si applica un passa-alto, al fine di eliminare le basse frequenze inutili dovute al rumore dei cavi o ai movimenti degli elettrodi. La frequenza di taglio storicamente usata si aggira tra i 5 e i 20 Hz, ma una frequenza più alta (50-60 Hz) può essere usata per migliorare la stabilità del controllo della protesi.

2.1.1 Analisi delle componenti indipendenti (ICA)

L'ICA è un metodo di elaborazione computazionale spesso usato per il segnale sEMG che permette di estrarre N sorgenti di segnali indipendenti tra loro da almeno N registrazioni (nel caso in esame, canali) che all'inizio della acquisizione sono linearmente dipendenti. Questo viene fatto mediante analisi statistiche multivarianti.
Teoricamente, avendo un numero sufficiente di canali, è possibile ridurre il cross-talking separando le attività dei singoli muscoli [13]. Questa procedura si applica quando vengono rilevati i burst del segnale, soprattutto nel caso dinamico (real-time).
Oltre alla separazione dei campioni, viene usato anche e soprattutto per ridurre il rumore del segnale.
L'algoritmo maggiormente utilizzato in ambito sEMG è il FastICA [1], per la sua semplicità, veloce convergenza e risultati soddisfacenti [14].

2.2 Analisi del segnale

2.2.1 Segmentazione dei dati

Una volta acquisito il segnale, questo deve essere segmentato per il calcolo delle caratteristiche. La segmentazione può avvenire sia dal segnale raw

[1] FastICA packages: `http://research.ics.aalto.fi/ica/fastica/`

EMG (range componenti principali 0-500 Hz) sia da quello rettificato (range componenti principali 0-50 Hz).
Quest'ultimo ($rect_{c,i}$), visibile in figura 2.2 in verde, viene calcolato nel modo seguente:

$$rect_{c,i} = |emg_{c,i} - M(emg_c)| \tag{2.2}$$

dove c è il canale e i è l'i-esimo campione del canale. Per $M(emg_c)$ si intende la media del segnale per il canale c.

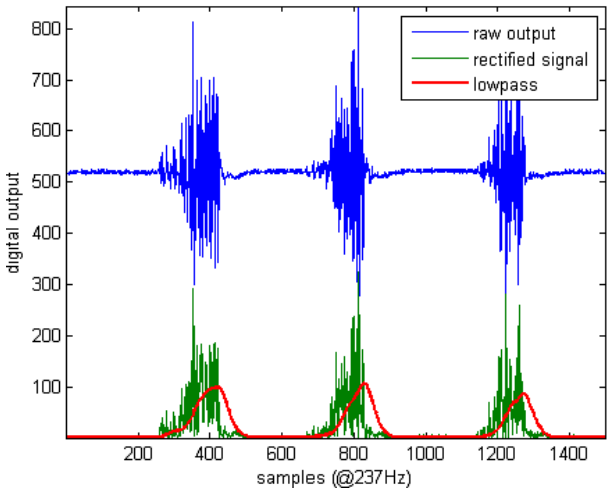


Figura 2.2: In verde il segnale rettificato, in rosso il segnale a seguito del passa-basso (si noti il ritardo per il processing)[13]

In entrambi i casi c'è la necessità di scegliere la finestratura adatta da applicare per la *rappresentazione delle caratteristiche*. Alla base ci sono due tecniche diverse di finestratura (Figura 2.3): quella adiacente e quella sovrapposta.

Nella prima si utilizzano dei segmenti disgiunti di segnale, mentre nella seconda si hanno dei segmenti sovrapposti. Nel caso di finestratura adiacente, la massima lunghezza di finestra deve essere inferiore ai 300 ms (requisito real-time) [15][16], mentre per quella sovrapposta il requisito deve essere soddisfatto per la lunghezza della sovrapposizione. In entrambi i casi si ha un ritardo τ di processing, che deve essere compreso all'interno dei 300 ms.
Englehart e Hudgins [15] hanno valutato l'effetto del segmento incrementale sulla performance di classificazione. Un segmento incrementale più piccolo produce un flusso di *class decision* più denso ma semi-ridondante, che può migliorare il tempo di risposta e l'accuratezza.
Nella tesi di Rivela e Scannella [1], viene prediletta la sovrapposizione, anche

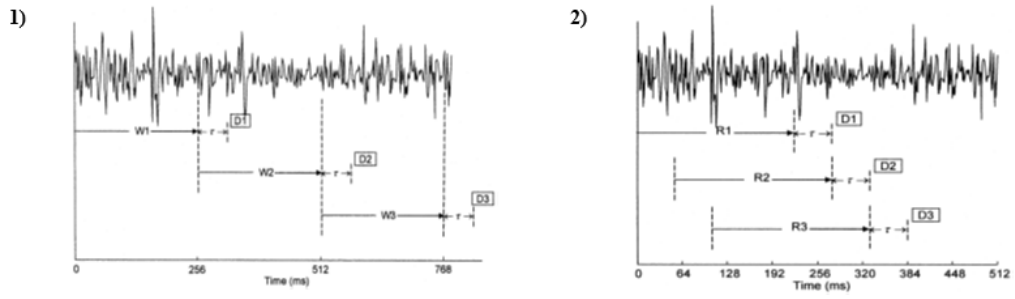


Figura 2.3: Tipologie di finestrature: **1)** adiacente, **2)** sovrapposta [15].

se altri studi [17] mostrano che, a seconda del tipo di caratteristiche valutate, la finestratura sovrapposta può non incidere sulle performance.

Scelto il tipo di finestratura, è importante definire la lunghezza di finestra. Una finestra troppo piccola può avere poca informazione, mentre una troppo grande aumenta i tempi di elaborazione.
Esistono diversi studi a proposito, sia per finestrature adiacenti che sovrapposte.
Englehart e Hudgins [15], per un segnale steady e per finestre adiacenti, propongono inizialmente una finestra di 256 ms fino a ridurla poi a 128 ms. Con l'utilizzo dello stato transiente invece la lunghezza rimane a 256 ms (essendo uno stato che presenta maggior rumore necessita di maggiore informazione).
Sempre con finestratura adiacente, Smith et Al.[18] valutano come ottimale finestre tra i 150 e i 250 ms, in seguito a un confronto con altre lunghezze.
Phinyomark et Al. [19], in uno studio più recente, valutano lunghezze superiori ai 250 ms, quindi, per soddisfare il requisito real-time (300 ms), adottano una finestratura sovrapposta. Valutano come ottimale una finestra di 500 ms, con un incremento di 125 ms.
Nella tesi di Rileva e Scannella [1], invece, viene identificata come lunghezza ottimale (sovrapposta), dopo una serie di prove e valutazioni su nove classi di riconoscimento, quella di 500 ms, con un incremento di 62 ms.

Oltre alla valutazione della finestra, bisogna scegliere quale stato del segnale utilizzare per il calcolo delle caratteristiche.
Difatti il segnale EMG è composto da uno stato transiente, che corrisponde alla contrazione isotonica del muscolo, e da uno stato di steady, che indica la contrazione isometrica. Rispettivamente vengono rilevati durante la fase iniziale del movimento e al raggiungimento della posizione target.
Diversi studi [6][20] dimostrano come lo stato di steady abbia un'accuratezza di classificazione maggiore rispetto allo stato transiente, soprattutto in fase

di training.
Nonostante questo, a partire da Hudgins [21], ci sono state diverse ricerche nella valutazione dello stato transiente, con lo scopo principale di anticipare il tempo di processing del segnale e quindi valutare il prima possibile la classe di appartenenza. Anticipare la decisione è utile durante una elaborazione in tempo reale, per cercare di rimanere al di sotto dei 300 ms richiesti in letteratura.
Inoltre studi come quello di Yang Et Al. [22] valutano la possibilità di usare entrambi gli stadi nella fase di training per migliorare la classificazione, specialmente per aumentare le performance durante il controllo dinamico del movimento.

2.2.2 Rappresentazione delle caratteristiche

La *rappresentazione delle caratteristiche* viene spesso definita come la componente che dà maggior peso all'accuratezza della classificazione.
Il segnale EMG in ingresso viene elaborato al fine di ottenere un'informazione utile ed eliminare i rumori e le parti del segnale che non sono di interesse. La rappresentazione dell'informazione, che verrà in seguito classificata, viene definita in due step:

- estrazione delle caratteristiche
- riduzione delle caratteristiche

2.2.2.1 Estrazione delle caratteristiche

Il segnale acquisito, in seguito a opportuni filtraggi, dovrà essere analizzato mediante una valutazione di caratteristiche calcolate.
Come già detto in precedenza, diversi studi [19] dimostrano che una buona scelta delle caratteristiche influenza in maniera significativa i risultati della classificazione; inoltre, scegliendo quelle appropriate, si possono evitare di implementare metodi di preprocessing [23] per limitare il rumore del segnale EMG. Difatti hanno lo scopo principale di enfatizzare le strutture invarianti del dato in ingresso, cercando di eliminare le informazioni irrilevanti (rumore).

Le varie caratteristiche devono essere valutate a seconda di diversi parametri, che ne definiscono la qualità [24] [25] [26] [23] [27]:

- Massima separabilità tra le classi

- Robustezza rispetto a varie condizioni di rumore (spostamento elettrodo, variazione contrazione, fatica)
- Complessità (per essere applicabili a sistemi real-time con hardware ragionevole)

In generale le tipologie di caratteristiche sono tre: caratteristiche nel dominio del tempo, nel dominio della frequenza e nel dominio tempo/frequenza [1].

Caratteristiche nel dominio del tempo

Questa tipologia di caratteristiche viene calcolata direttamente sul segnale senza effettuarne delle trasformazioni e ha quindi una bassa complessità computazionale. Tali caratteristiche però hanno lo svantaggio di presupporre che il segnale sia stazionario [28], quindi sono meno performanti per segnali dinamici.
Di seguito verranno descritte quelle maggiormente usate oggi [29] [30] [1].

- **Root Mean Square (RMS)**

  É lo scarto quadratico medio che definisce la potenza media del segnale.

$$RMS = \sqrt{\frac{1}{N}\sum_{i=1}^{N} x_i^2} \tag{2.3}$$

- **Zero Crossing (ZC)**

  Conta il numero di volte che il segnale passa lo zero, così da dare una stima base della frequenza di contrazione.

$$ZC = \sum_{i=1}^{N-1} [sgn(x_i X x_{i+1}) \cap |x_i - x_{i+1}| \geq soglia] \tag{2.4}$$

$$sgn(x) = \begin{cases} 1, & \text{se } x \geq soglia \\ 0, & \text{altrimenti} \end{cases}$$

- **Mean Absolute Vale (MAV)**

  Simile a RMS, può anche essere usato per valutare l'intensità di una

contrazione. È la caratteristica più popolare nell'analisi del segnale EMG.

$$MAV = \frac{1}{N}\sum_{i=1}^{N}|x_i| \tag{2.5}$$

- **Slope Sign Changes (SSC)**

  Misura le caratteristiche di frequenza del segnale contando il numero di volte che la pendenza del segnale cambia. Simile allo ZC, viene applicata una soglia per ridurre il rumore.

$$SSC = \sum_{i=2}^{N-1}[f(x_i - x_{i-1})X(x_i - x_{i+1})] \tag{2.6}$$

$$f = \begin{cases} 1, & \text{se } x \geq soglia \\ 0, & \text{altrimenti} \end{cases}$$

- **Variance (VAR)**

  La varianza quadratica o deviazione standard di un segnale EMG, assumendo la media pari a zero, dà una stima della potenza del segnale.

$$VAR = \frac{1}{N-1}\sum_{i=1}^{N}x_i^2 \tag{2.7}$$

- **Willison Amplitude (WAMP)**

  É un altro modo per valutare la forza durante uno sforzo muscolare. Viene contato il numero di volte che la differenza tra due campioni consecutivi eccede una soglia prefissata in una finestra di tempo scelta.

$$WAMP = \sum_{i=1}^{N-1}[f(x_i - x_{i+1})] \tag{2.8}$$

$$f = \begin{cases} 1, & \text{se } x \geq soglia \\ 0, & \text{altrimenti} \end{cases}$$

- **Waveform Length (WL)**

  Estrae la lunghezza di una waveform in una certa finestra, in modo

tale da fornire informazioni su ampiezza e variazione di frequenza di un segnale mioelettrico.

$$WL = \sum_{i=1}^{N-1} |x_{i+1} - x_i| \tag{2.9}$$

- **Myopulse Percentage Rate (MYOP)**

  Il valore medio di Myopulse Output è pari a 1 quando il valore assoluto del segnale EMG supera una certa soglia prefissata. Contiene informazioni sulla frequenza del segnale.

$$MYOP = \frac{1}{N} \sum_{i=1}^{N} f(x_i) \tag{2.10}$$

$$f(x) = \begin{cases} 1, & \text{se } x \geq soglia \\ 0, & \text{altrimenti} \end{cases}$$

- **Auto-regressive Coefficients (AR)**

  I coefficienti di regressione sono un modello di predizione in cui i campioni del segnale vengono stimati da una combinazione lineare dei precedenti, più un termine di errore w_i(rumore bianco). L'ordine P del modello viene valutato in modo tale che un suo aumento provochi un aumento insignificante del rumore. Diversi studi [31] consigliano il modello del IV ordine.

$$x_i = \sum_{p=1}^{P} a_p x_{i-p} + w_i \tag{2.11}$$

- **Cepstral Coefficient (CC)**

  Vengono utilizzati soprattutto come caratteristiche del linguaggio. Vengono ricavati dai coefficienti AR e costituiscono un modello di predizione. È stato però provato che sono molto utili anche nella classificazione del movimento [24].

$$\begin{aligned} c_1 &= -a_1; \\ c_p &= -a_p - \sum_{l=1}^{p-1} \left(1 - \frac{l}{p}\right) a_p c_{p-l} \end{aligned} \tag{2.12}$$

- **Sample Entropy (SampEn)**

  Estrae l'informazione di similarità nelle serie nel tempo. Deriva dalla caratteristica poco utilizzata ApEn, ma evita gli errori dovuti al self-matching. È il logaritmo naturale negativo di una stima della probabilità condizionale che, se i modelli di serie di dati sono simili per un fattore m all'interno di un valore di tolleranza r, allora rimarranno tali al successivo punto di comparazione [19]. É la più popolare nell'analisi del segnale EMG.

$$SampEn = -ln\left(\frac{A^m(r)}{B^m(r)}\right) \tag{2.13}$$

Tra le caratteristiche nel dominio del tempo si ha la possibilità in generale di ricavare quattro tipi di informazioni diverse, e per ognuna di queste è stata valutata la caratteristica migliore per *accuratezza di classificazione* (evitando così inutili ridondanze dovute all'uso di più caratteristiche dello stesso set):

- Informazione sull'intensità del segnale : MAV
- Informazione sulla complessità del segnale: WL
- Informazione sulla frequenza: WAMP
- Modello di predizione: AR

Esiste un'altra tipologia di caratteristiche nel dominio tempo- dipendenti (MAVS,HEMG, ..), che derivano da quelle relative all'intensità del segnale. Queste non sono state descritte poiché sono state valutate inadatte alla classificazione del segnale EMG [1].

Secondo alcuni studi [19] è stato riscontrato che l'utilizzo di SampEn, presa anche singolarmente, di per sé porta a un generoso aumento delle performance.
Rispetto alla robustezza al rumore, tra le caratteristiche citate alcuni articoli definiscono come migliori rispetto alla sensitività ZC e AR del IV ordine [19], mentre in generale WL e WAMP per qualità di riconoscimento in un ambiente rumoroso [23].
Tali valutazioni sono prese sulle singole caratteristiche, mentre per una valutazione su un set di caratteristiche si rimanda nella sezione dedicata 2.2.2.2.

Caratteristiche nel dominio della frequenza

Questa tipologia di caratteristiche valuta maggiormente la fatica del muscolo e l'analisi del reclutamento delle unità motorie. Esempi di caratteristiche sono Mean Frequency(MNF) e Median Frequency (MDF), derivanti dall'analisi di densità spettrale di potenza (PSD).
Utilizzano solitamente la trasformata di Fourier (Equazione 2.14), perdendo però informazioni del segnale nel dominio del tempo.
Non verranno descritte nel dettaglio poiché, oltre a non essere particolarmente usate nella classificazione, è stato dimostrato che hanno basse performance nella separabilità tra classi, e quindi sono svantaggiose dal punto di vista della classificazione di un movimento. Quelle con performance più elevate non hanno comunque risultati migliori rispetto a quelle nel dominio del tempo, che hanno complessità inferiore.

Caratteristiche nel dominio del tempo-frequenza

Un compromesso tra le caratteristiche del tempo e quelle della frequenza si trova in questa famiglia, per molti la più adatta al segnale EMG.
Queste caratteristiche creano una rappresentazione tempo-frequenza del segnale, combinando l'analisi nel dominio del tempo e quella nel dominio della frequenza finora analizzate per avere una localizzazione temporale delle caratteristiche spettrali di un segnale [16].
Essendo l'EMG un segnale non stazionario, questa rappresentazione comporta un aumento di informazione utile ai fini dell'analisi del segnale.
Uno svantaggio può essere la loro alta dimensionalità, che però può essere migliorata mediante la *riduzione delle caratteristiche* (vedi il paragrafo dedicato).

Possono essere divisi in due gruppi: i metodi lineari e quelli quadratici. Questi ultimi non verranno presi in considerazione, principalmente perché presentano un'alta complessità, rendendoli non utilizzabili per sistemi real-time.
Tra i metodi lineari, i principali tipi di rappresentazione per valutare caratteristiche del tempo-frequenza sono i seguenti [16] [1][32]:

- Short-time Fourier transform
- Wavelet transform
- Wavelet packet transform

Rappresentano l'informazione in un piano tempo-frequenza e si differenziano a seconda delle modalità di partizionamento dello spazio (Figura 2.4).

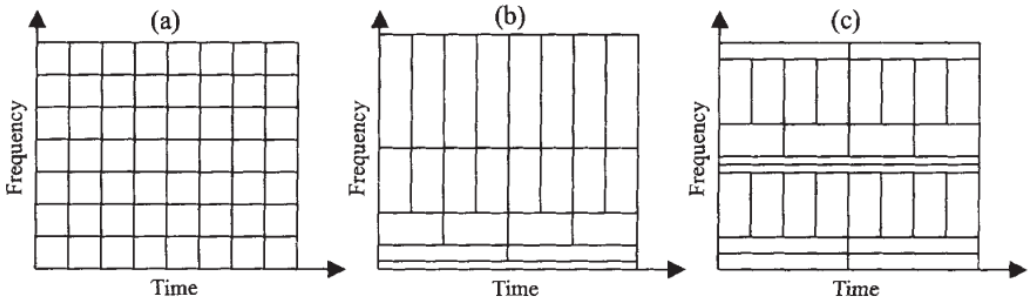


Figura 2.4: Metodi lineari per il calcolo delle caratteristiche nel dominio del tempo-frequenza. **a)** Short-Time Fourier Transform **b)** Wavelet Transform **c)** Wavelet Packet Transform [32]

Short-time Fourier transform (STFT) Questa rappresentazione ha una partizione fissa (*aspect ratio*): una volta specificata, ciascuna cella del piano ha le stesse dimensioni in termini temporali e di frequenza.
Si basa sulla trasformata di Fourier, introducendo però il concetto di *local frequency* per aggiungere informazioni temporali. Normalmente la trasformata di Fourier

$$X(f) = \int_{-\infty}^{\infty} x(t) e^{2j\pi ft} dt \tag{2.14}$$

denota la distribuzione del segnale nel dominio della frequenza per l'intero campione, senza una risoluzione temporale. Questo può andare bene per un segnale stazionario, ma se ci dovessero essere dei bruschi cambiamenti dovuti alla non stazionarietà, questi si propagherebbero per l'intero spazio.
Introducendo il concetto di *local frequency*, la trasformata effettuerà una valutazione all'interno di una finestra temporale definita.
É importante la scelta della risoluzione temporale e della frequenza, visto che sarà fissa per tutto il piano, e questo è il maggior svantaggio di questo tipo di rappresentazione. Difatti generalmente la distribuzione dell'energia del segnale non è la stessa per ogni parte del piano.

Wavelet transform (WT) È uno strumento che permette di scomporre il segnale in diverse componenti frequenziali, e quindi studiare ogni componente con un risoluzione adatta alla sua scala [16] [33]. Ciò che ne risulta è che la trasformata di un segnale che evolve nel tempo dipende da due variabili: la scala (o frequenza) e il tempo. In questo tipo di rappresentazione la risoluzione temporale e di frequenza variano all'interno del piano, anche se la partizione rimane fissa rispetto al segnale.

Questa tipologia di caratteristiche viene usata soprattutto quando il segnale non è stazionario e, nel caso dell'EMG, quando viene valutato lo stato transiente.

Esistono 2 tipologie di Wavelet transform: quella continua (CWT) e quella discreta (DWT). La *continuous wavelet transform* (CWT) definisce la trasformata come:

$$CWT_x(\tau, a) = \frac{1}{\sqrt{a}} \int x(t) \Psi^* \left(\frac{t-\tau}{a} \right) dt \tag{2.15}$$

dove $\Psi(t)$ è una finestra-prototipo chiamata *mother wavelet* o *wavelet function* (WF). L'analisi determina una correlazione del segnale con una versione *shiftata* (di τ) e *scalata* (di a) rispetto alla frequenza della WF.
Di conseguenza la risoluzione in frequenza sarà proporzionale alla frequenza centrale (la mother wavelet appunto). La *discrete wavelet transform* (DWT) ha una complessità computazionale inferiore, e quindi è preferibile alla CWT per applicazioni real-time [34]. É definita come la CWT (Equazione 2.15), ma i due parametri τ e a assumono unicamente valori discreti.
In sostanza, trasforma iterativamente il segnale di interesse in più sottoinsiemi multi-risoluzione di coefficienti. Tale procedimento è denominato *Wavelet Decomposition.*
Ciò avviene filtrando il segnale sEMG con un filtro passa-alto e un filtro passa-basso, i cui coefficienti dipendono dal tipo di *wavelet function* (WF) utilizzata. Vengono così generati rispettivamente, a partire dal segnale sEMG, un sottoinsieme di coefficienti di dettaglio (cD1) e un sottoinsieme di coefficienti di approssimazione (cA1) al primo livello. Per ottenere un'analisi multi-risoluzione, viene eseguito questo procedimento per più livelli, fino a raggiungere il livello finale desiderato (Figura 2.5). Nell'analisi EMG, si è osservato che la performance migliore si ottiene con quattro livelli di scomposizione, [35] [36] [37].

Ciascun sottoinsieme di coefficienti wavelet può essere poi ricostruito, utilizzando la DWT inversa (IDWT), per stimare una componente efficace del segnale sEMG. La IDWT è calcolata usando i coefficienti di tutte le componenti wavelet. Quindi a partire dai coefficienti cA4 e cD1, cD2, cD3, cD4 si possono ottenere i relativi segnali EMG.
Diversi studi [38] [34] dimostrano come sia preferibile selezionare solo i coefficienti di determinate frequenze invece che prelevare il sottoinsieme totale. Questo in modo tale da avere un'accuratezza simile ma limitare il costo computazionale.

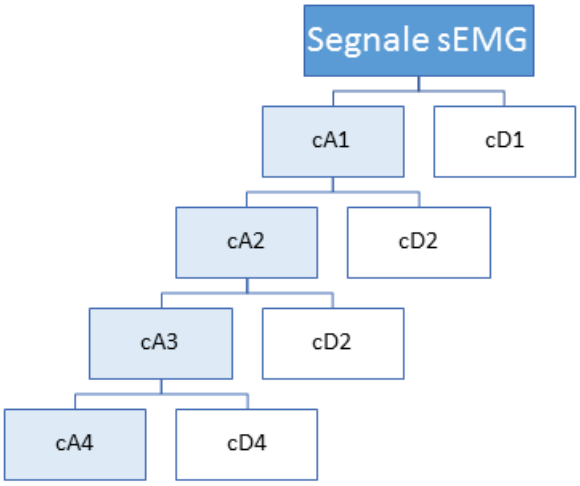


Figura 2.5: Procedimento Discrete Wavelet Transform

In particolare si è osservato che la miglior performance di classificazione può essere ottenuta usando:

- il segnale ricostruito dai coefficienti di dettaglio al livello 2 (D2), scegliendo come WF la Daubechies wavelet di ordine 7 (db7) e calcolando in seguito la caratteristica ZC;
- il segnale ricostruito dai coefficienti di dettaglio al livello 1 (D1), scegliendo come WF la Daubechies wavelet di ordine 5 (db5) e calcolando in seguito la caratteristica MYOP.

Nello stesso studio si è mostrato come l'adeguatezza di un sottoinsieme di coefficienti e di una WF dipenda dal tipo di caratteristica che si vuole estrarre [1].

Wavelet packet transform (WPT) Questa rappresentazione permette una risoluzione tempo-frequenza adattiva [16] e costituisce una generalizzazione della wavelet decomposition (DWT). Oltre a decomporre le approssimazioni del segnale, vengono presi in considerazione anche i dettagli.
Come risultato, la risoluzione del piano tempo-frequenza è configurabile: il partizionamento dell'asse delle frequenze potrà cambiare a seconda della necessità dell'applicazione.
Si basa sul *Best-Basis Paradigm* [39], che si definisce attraverso tre passi:

1 Selezionare una base *migliore* per il problema scegliendo tra una libreria di basi (ottenute nella WPT mediante la decomposizione).

2 Si ordinano le caratteristiche per *importanza* rispetto al problema, scartando quelle meno significative.

3 Si utilizzano le caratteristiche rimaste per risolvere il problema.

La definizione di *migliore* e più *importante* dipende dal tipo di problema. Nella classificazione bisogna cercare di ottenere il minor numero di caratteristiche, cercando di mantenere la massima separabilità delle classi.

Iniziando da una finestra del segnale EMG, il primo livello di decomposizione genera due sottobande passa-alto e passa-basso (come nella WT), ognuna di lunghezza dimezzata rispetto al segnale in ingresso.
Il secondo stadio di decomposizione genererà invece quattro sotto-sequenze del segnale, ancora con una lunghezza dimezzata rispetto al precedente livello.
Questo processo crea una struttura ad albero binario di wavelet packet (Figura 2.6), dove i nodi dell'albero rappresentano i sottospazi con le diverse caratteristiche di localizzazione della frequenza.

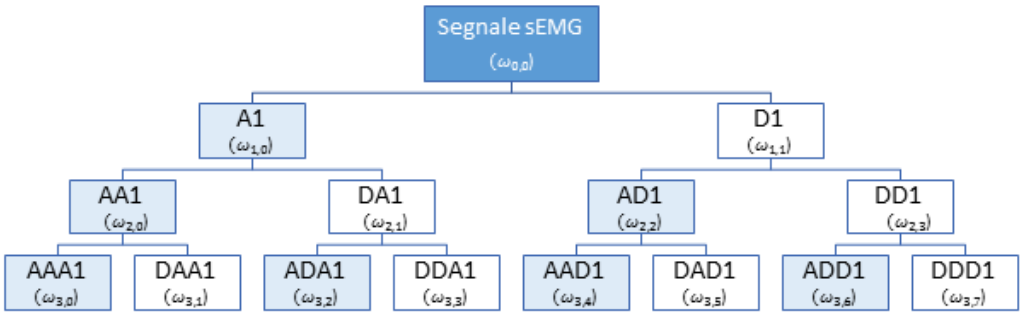


Figura 2.6: Procedimento Wavelet Packet Transform

La radice dell'albero è $\omega_{0,0}$ e rappresenta il segnale originario.
Un sottospazio $\omega_{j,k}$ è decomposto in due sottospazi ortogonali

$$A : \omega_{j,k} \rightarrow \omega_{j+1,2k} e D : \omega_{j,k} \rightarrow \omega_{j+1,2k+1} \tag{2.16}$$

dove j definisce la scala e k l'indice di sottobanda all'interno della scala. Questo può essere espresso come $\omega_{j,k} = \omega_{j+1,2k} \bigoplus \omega_{j+1,2k+1}$ per $j = 0, ..., J$ e $k = o, ..., 2^j - 1$ che genera l'intera decomposizione per la scala J.

La WPT permette, quindi, di rappresentare il segnale originario utilizzando varie combinazioni di componenti ad alta e bassa frequenza. Per un livello di scomposizione pari a n, si avranno $2^{2^{n-1}}$ possibili modi per descrivere il segnale originario. Ad esempio con n = 3 avremo 16 possibili modi per descrivere il segnale $\omega_{0,0}$.
Ne deriva che il set di sottospazi nell'albero binario WPT è ridondante, cioè la trasformata fornisce un albero binario di coefficienti comprendendo 2^N possibili basi ortonormali, dove N è la lunghezza di registrazione. La potenza della WPT è che la base migliore può essere scelta per un'applicazione specifica, identificandola da un'insieme di possibili candidate mediante una funzione costo. L'algoritmo di selezione della base migliore ha origine nella compressione del segnale, e tutte le funzioni costo associate con la compressione implicano una certa misura di entropia. Questo può essere considerato ottimale per la compressione del segnale, ma potrebbe essere inappropriato per la classificazione.
Saito e Coifman [39] hanno descritto un metodo per definire le *basi migliori*, chiamato *Local Discriminant Bases* (LDB).
Tale algoritmo ha lo scopo di selezionare dall'albero binario la base ortonormale che più discrimina il set di classi in esame. La misura di separabilità delle classi viene definita mediante la misura di discriminazione D e, tra i diversi possibili metodi per calcolarla [16], è stata valutata come migliore in campo di classificazione l'*entropia simmetrica relativa*:

$$D(p,q) = \sum_{i=1}^{n} p_i log\frac{p_i}{q_i} + \sum_{i=1}^{n} q_i log\frac{q_i}{p_i} \tag{2.17}$$

dove n rappresenta il numero di caratteristiche, p e q sono invece due diverse classi da discriminare.
Per ottimizzare la rappresentazione, i parametri in ingresso a D (nella formula generale p e q) sono sostituiti dalle mappe di energia tempo-frequenza Γ^k per ogni classe k, che indicano la distribuzione di energia della classe nel piano, calcolato rispetto a un set di segnali di training della relativa classe.
Quindi, partendo dall'albero binario creato dalla decomposizione, vengono calcolate le mappe di energia-frequenza e, attraverso la discriminazione D, vengono via via eliminate le basi con pesi minori [39].

2.2.2.2 Insiemi di caratteristiche e loro riduzione

É chiaro che l'informazione ottenuta dall'analisi del segnale non dipenda dalla singola caratteristica, ma dall'insieme delle caratteristiche valutate sul segnale.

É già stato specificato che spesso caratteristiche diverse hanno informazioni ridondanti tra loro, per cui la scelta del set migliore è fondamentale per ottenere buone performance.
Uno dei primi insiemi di caratteristiche, e anche quello più utilizzato, è il set di Hudgins. Comprende il calcolo di quattro caratteristiche: MAV, SSC, WL, ZC. Questo set viene usato soprattutto nei classificatori che utilizzano la fase di steady e appartiene alla categoria delle caratteristiche nel tempo.
Nello studio di Rivela e Scannella [1], viene valutato questo set (che è il rappresentante dei set di caratteristiche nel tempo) come uno di quelli con migliori performance impiegando un classificatore LDA. Inoltre vengono definiti e testati anche un set definito da Phinomark et al. [19], caratterizzato dalla sua robustezza nel tempo, uno definito dell'articolo [34], che utilizza anche le wavelet, e infine il set di caratteristiche costituito dai coefficienti delle WPT (Tabella 2.1).

Compensando con altri studi [14] [25], si può affermare che in generale le WT e soprattutto le WPT hanno performance migliori, soprattutto in caso di stato transiente. Ad oggi, però, il set di Hudgins rimane ancora quello maggiormente utilizzato, perché ritenuto abbastanza discriminatorio e con un buon rapporto performance-complessità [25] [11] [6].

Riduzione delle caratteristiche

Il calcolo delle caratteristiche spesso presenta un'alta dimensionalità, soprattutto in quelle del dominio tempo-frequenza oppure in una situazione multicanale.
É indispensabile, soprattutto nelle applicazioni reali, cercare di limitarne il numero, mantenendo il più possibile invariata la potenza espressiva dell'informazione.
Sono stati quindi ideati metodi di riduzione della dimensionalità (DR) attraverso delle tecniche di *feature mapping*, al fine di ridurre le variazioni intra-classe e mantenere o esaltare quelle inter-classe, nonché limitare la ridondanza di informazione.
Tramite la DR si rappresenta quindi un dato multidimensionale in una dimensione ridotta ma comunque significativa. Tale rappresentazione avrà idealmente dimensionalità corrispondente a quella intrinseca dei dati, che rappresenta il minimo numero di parametri necessari per descrivere le proprietà dei dati osservati e che viene quindi scelta come dimensionalità appropriata dello spazio ridotto delle caratteristiche.
Quindi, assumendo che il dataset sia rappresentato da una matrice X, $n \times D$, formata da n vettori con dimensionalità D, e che il dataset abbia dimensio-

Nome del set	Caratteristiche presenti	Dominio
Hudgins	▪ MAV ▪ SSC ▪ WL ▪ ZC	Tempo
Phinyomark et al. (con SampEn)	▪ SampEn ▪ CC di ordine 4 ▪ RMS ▪ WL	Tempo
Phiniomark et al. (con wavelet)	▪ ZC ▪ WAMP ▪ MAV (con segnale ricostruito dai coefficienti di II livello della wavelet madre Daubechies di ordine 7) ▪ MYOP (con segnale ricostruito dai coefficienti del I livello della wavelet madre Daubechies di ordine 5)	Tempo-Frequenza
WPT	▪ Coefficienti della WPT (utilizzando come wavelet function Symmlet di ordine 5 (sym5) e un livello di decomposizione di 4)	Tempo-Frequenza

Tabella 2.1: Set di caratteristiche a confronto [1].

nalità d (con $d \ll D$) con dimensionalità intrinseca, si indica che i punti del dataset X giacciono su un dataset di dimensionalità d che è interno allo

spazio D-dimensionale. La DR trasforma perciò un dataset X con dimensionalità D in un nuovo dataset Y di dimensionalità d, cercando di conservare il più possibile la geometria dei dati.
Perciò, la dimensionalità intrinseca determina essenzialmente se il dato pattern D-dimensionale possa essere descritto adeguatamente in un sottospazio di dimensionalità minore di D.
La tecnica della DR è, quindi, in grado di ridurre la dimensionalità dei dati mantenendo la capacità di discriminazione, riducendo la quantità di memoria richiesta e aumentando la velocità del classificatore grazie alla riduzione del numero di parametri di apprendimento necessari. Inoltre riduce il problema della *curse of dimensionality* che si traduce nel peggioramento delle performance di classificazione che si ha quando il numero di campioni di training utilizzati nel progetto del classificatore è piccolo rispetto al numero delle caratteristiche. Bisogna, comunque, ricordare che un'eccessiva riduzione del loro numero può condurre ad una perdita del potere di discriminazione e quindi ad una minore accuratezza del sistema di riconoscimento risultante [3] [1].

Esistono due metodi per effettuare la DR: la *selezione delle caratteristiche* e la *proiezione delle caratteristiche*.
Il primo utilizza le caratteristiche calcolate e le ridimensiona, cercando di selezionare solo un sottoinsieme ottimale delle caratteristiche preesistenti.
Il secondo invece genera un nuovo insieme di caratteristiche, basandosi su una mappatura di quelle iniziali.
La scelta di un metodo rispetto all'altro dipende dal tipo di classificazione e dal tipo di dato che si sta utilizzando: se l'informazione discriminante tende a concentrarsi in poche caratteristiche nello spazio originale, allora sarà indicato utilizzare la *selezione delle caratteristiche*, se invece i dati in ingresso sono fortemente dinamici e si ha spesso una dispersione dell'informazione (come nel caso dell'sEMG) allora si predilige la *proiezione delle caratteristiche*.
Di seguito verrà valutata solo quest'ultima tecnica, poiché è quella utilizzata per il tipo di segnale in esame.

Proiezione delle caratteristiche

Nella *proiezione delle caratteristiche* una funzione effettua un mapping delle caratteristiche tra il set iniziale e un set di dimensioni inferiori.
Tale funzione può essere lineare o non lineare.

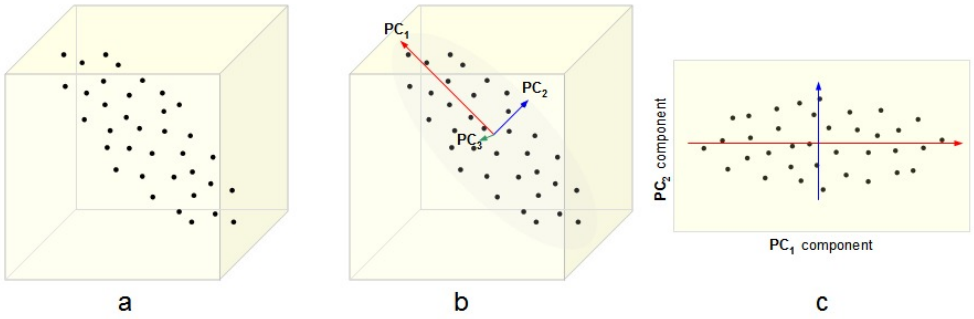


Figura 2.7: Illustrazione PCA. **a)** Dataset tridimensionale. **b)** I tre Principal Components (PCs) ortogonali per i dati, ordinati rispetto alla varianza. **c)** Proiezione del dataset nelle prime due PCs, scartando la terza. [40]

Tecniche lineari: Analisi delle componenti principali (PCA) La PCA è la tecnica più diffusa tra quelle lineari: cerca il sistema di coordinate che spiegano meglio la varianza dei dati e si basa sull'errore quadratico medio. Infatti provvede ad effettuare un mapping lineare che minimizzi questo errore, eliminando le dipendenze lineari e il rumore nei dati [16].
È una tecnica non supervisionata, motivo per cui non è finalizzata a ottimizzare il problema di classificazione, ma semplicemente viene usata per la DR. Essa riduce la dimensionalità selezionando un nuovo spazio di caratteristiche, scegliendo gli autovettori che hanno maggiore autovalore (Figura 2.7). Essenzialmente questo mantiene le parti che hanno varianza maggiore, e quindi solitamente quelle che hanno un'informazione più discriminante.
La principale limitazione della PCA è che non considera la separabilità delle classi, poiché non mantiene informazioni sulle tipologie di caratteristiche originarie. Semplicemente proietta il set di partenza in un spazio i cui assi rappresentano le direzioni di massima varianza, ma non è garantito che tali assi siano quelli che discriminano maggiormente l'informazione.
Altre tecniche lineari che non verranno descritte perché non usate in questo studio di tesi e perché sono impiegate in maniera minore in casi reali sono la LDA (Linear Discriminant Analysis) e l'ULDA (Uncorrelated Linear Discriminant Analysis).

Tecniche non lineari In generale, le principali tecniche non lineari utilizzate in letteratura sono le seguenti [3]:

- Multidimensional Scaling (MDS) [41]

Rappresenta un dataset multi-dimensionale in due o tre dimensioni in modo tale che la matrice delle distanze nello spazio originale venga preservata il più possibile nello spazio proiettato. Il problema principale del MDS è che non viene esplicitata una funzione di mapping univoca, per cui se si vuole introdurre un nuovo pattern nella mappa di un dato training set è necessario ripetere nuovamente il mapping.

- Self-Organized Feature Map (SOFM) o Kohonen Map [42]
  Si basa su una rete neurale in cui i neuroni sono disposti in una griglia n-dimensionale, dove n può essere 1, 2 (caso più usato) o 3. Nella sua forma base produce un grafo di similarità di dati in input. É la tecnica non lineare maggiormente usata in ambito EMG, tuttavia spesso la si utilizza in termini di classificatore. Verrà pertanto descritta maggiormente al paragrafo corrispondente nella sezione *Classificazione*.

.

Tecniche ibride Come già detto in precedenza, la PCA non mantiene l'informazione della separabilità delle classi. Per migliorare questa tecnica, nell'articolo di Chu Et Al. [43] hanno ideato un metodo ibrido che affianca la PCA lineare con la SOFM non lineare, che è stato poi utilizzato in diversi studi. La DR svolta dalla PCA riduce la complessità per il classificatore e quindi il tempo di processing, mentre il mapping non lineare della SOFM trasforma le caratteristiche ridotte dalla PCA in uno spazio con maggiore separabilità delle classi. Il classificatore quindi troverà un iperpiano con un margine di separazione esaltato.

2.3 Classificazione

Lo scopo del riconoscimento di un pattern nel segnale EMG é la classificazione, ovvero l'associazione di una classe ad uno specifico comportamento del segnale in ingresso. Nel problema in esame la finalità della classificazione é quella di identificare l'intenzione di movimento, quindi associare la classe di movimento corretta all'attività muscolare residua dell'utente che utilizza la protesi.

In letteratura la classificazione si divide in due metodi principali: l'apprendimento Supervisionato e l'apprendimento Non Supervisionato. Il primo approccio necessita di una conoscenza a priori della distribuzione di dati sulla quale il sistema di classificazione dovrà operare, mentre nell'apprendimento non supervisionato non sempre é possibile disporre di queste informazioni [44].

Dato un pattern è quindi possibile effettuare due operazioni [2]

- Identificare l'appartenenza del pattern ad una classe predefinita *(Discriminazione)*.
- Identificazione del pattern come appartenente ad una classe sconosciuta a priori *(Raggruppamento)*.

Di seguito verranno esposti principalmente i classificatori supervisionati essendo i più indicati per il problema trattato.

2.3.1 Diversi Tipi di Classificatori

Gli approcci maggiormente utilizzati nella Pattern Recognition sono i seguenti: Template Matching, Neural Network, Statistico, Sintattico/Strutturale.

Template

Il Template Matching cerca di associare un pattern non classificato (Test Pattern) ad un elemento a lui simile appartenente ad un insieme di riferimento (Templates Set). I template, se parliamo di Visione Artificiale, sono degli oggetti campione all'interno di una scena; possono essere particolari costrutti del linguaggio, se parliamo di analisi di un testo. Questo approccio cerca di associare il dato da classificare ad un costrutto di riferimento piuttosto che ad una classe astratta.
Questa tecnica é particolarmente indicata laddove esiste un modello a cui riferirsi piuttosto che un insieme di dati raggruppati in insiemi di appartenenza.

I pattern da associare vengono rappresentati prevalentemente sotto forma di campioni se si tratta di un segnale, sotto forma di pixel per l'analisi di video o di immagini o sotto forma di caratteri e simboli se vengono analizzati dei testi [3], ma esistono in letteratura degli esempi dove il pattern da classificare viene rappresentato attraverso un vettore di caratteristiche,analogamente a quello che succede per le altre tecniche, e non come dato grezzo [44]. Come parametro di riconoscimento viene utilizzata una funzione di costo, che spesso é espressa sotto forma di distanza nello spazio delle caratteristiche o dei campioni tra il Test Pattern e il Template. Come funzioni di distanza in letteratura vengono spesso utilizzate: la distanza Euclidea, la norma di Frobenius oppure la Distanza di Modifica [44]. In altri casi viene invece ritenuta più utile la funzione di correlazione [3] [44], la quale meglio si applica a problemi rappresentati da campioni al posto di vettori di caratteristiche.

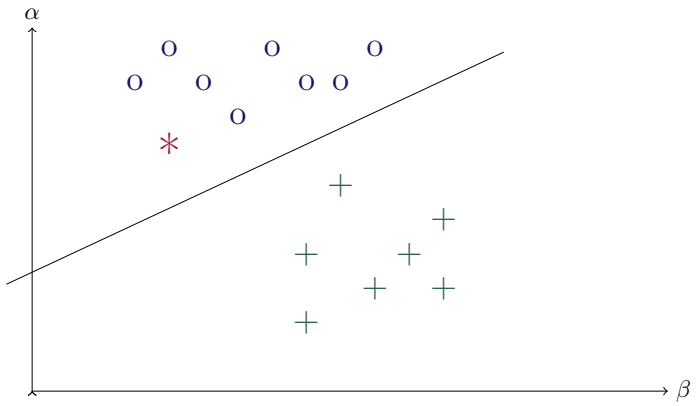


Figura 2.8: I soggetti appartenenti alla classe A sono rappresentati con o mentre i soggetti appartenenti alla classe B sono reppresentati con +, e * rappresenta il nuovo soggetto da classificare

Statistico

L'approccio statistico cerca di separare i pattern trovando un confine tra le classi e scegliendo delle caratteristiche di rappresentazione che massimizzino la variabilità inter-classe e minimizzando la variabilità intra-classe. Ogni pattern è un punto in uno spazio d-Dimensionale, ed è costituito da d caratteristiche. Lo scopo di queste proprietà è di far occupare ai vettori di caratteristiche appartenenti alla stessa classe uno spazio compatto e distante dall'area occupata dalle altre classi[1]. Avendo selezionato le caratteristiche per minimizzare la variabilità intra-classe, i pattern appartenenti alla stessa classe occuperanno un area circoscritta nello spazio d-dimensionale: quindi come è mostrato in figura-2.8 è ragionevole sostenere che il nuovo pattern (*) apparterrà più probabilmente alla classe identificata da 0 piuttosto che alla classe identificata da +.

Selezionando le caratteristiche aventi anche la seconda proprietà (distanza inter-classe massima) sarà ragionevole sostenere che la sovrapposizione delle classi sarà minima [44]. Queste proprietà ci consentono di scegliere delle caratteristiche che sviluppano dei confini decisionali che separino correttamente e con maggior margine le classi.
Nello *Statistical Decision Theoretic Approach*, i confini decisionali vengono definiti in base alle classi stesse: le quali devono essere apprese per mezzo del training set.

Nella Discriminant Analysis-Based Approach, i confini vengono definiti in modo parametrico a priori, scegliendo la complessità del modello (lineare,

quadratico, ecc.). I parametri del modello vengono poi appresi mediante le tecniche di *machine learning* [1].

Sintattico - Strutturale

Molti problemi di riconoscimento richiedono di analizzare pattern complessi. In questi casi fornisce risultati migliori l'approccio gerarchico, dove un pattern può essere visto come composto da pattern più semplici che sono costituiti a loro volta da pattern elementari [45] [46]. Le più piccole unità riconosciute sono le primitive, che vengono poi aggregate con regole sintattiche per comporre il pattern più complesso che si vuole riconoscere. L'identificazione avviene esplorando le regole sintattiche componendo le primitive identificate.

L'approccio sintattico può portare alla costruzione di uno spazio di ricerca molto grande che ha crescita combinatoria, il quale richiede un grande training set e un sforzo computazionale molto intenso [47].

Reti Neurali

Le reti neurali sono dei sistemi composti da unità computazionali elementari chiamate neuroni, interconnesse tra di loro per formare una rete che collabora per il calcolo del risultato finale. Le reti neurali in letteratura sono presenti in varie topologie, ma tutte hanno delle caratteristiche in comune: sono formate da un insieme di unità computazionali organizzate in un grafo pesato, dove i nodi sono i neuroni e le connessioni tra i neuroni sono gli archi pesati che connettono l'input all'output attraverso la rete.
Le reti neurali hanno la capacità di imparare complesse relazioni non lineari di input-output, utilizzando una procedura di apprendimento sequenziale per adattarsi alla specifica distribuzione di dati[3]. Quelle più utilizzate per la classificazione dei pattern sono le feed-forward [48], di cui due esempi sono il Perceptrone Multi Livello (MLP) e la Radial-Basis-Function (RBF). Queste reti sono organizzate in livelli successivi connessi unidirezionalmente dall'ingresso verso l'uscita.

Un altro tipo di reti neurali usate in letteratura sono le reti di Kohonen, più comunemente chiamate Self Organizing Map (SOM), le quali vengono impiegate per creare corrispondenze non lineari nello spazio delle caratteristiche (come già visto in precedenza) o per apprendimento non supervisionato per mezzo del raggruppamento.

Il processo di apprendimento delle reti neurali consiste nell'apportare modifiche ai pesi del grafo, i quali interagiscono sulle relazioni d'ingresso e d'uscita; questa operazione di apprendimento viene generalmente effettuata uti-

lizzando i dati di training. Il frequente impiego delle reti neurali nei problemi di identificazione è dovuto alla loro scarsa dipendenza dalle informazioni sul dominio applicativo e alla loro elevata velocità nel processare un campione in ingresso una volta effettuato l'apprendimento [3].

Breve comparazione dei classificatori finora descritti

Approccio	Rappresentazione	Funzione di Riconoscimento	Criterio
Template	Campioni	Correlazione	Err Classificazione
Statistico	Caratteristiche	Discriminazione	Err Classificazione
Strutturale	Primitive	Grammatiche	Err Accettazione
Reti Neurali	Campioni, Caratteristiche	Discriminazione di Rete	Err Quadratico

Tabella 2.2: Tipologie di classificatori maggiormente utilizzati per il Pattern Matching [3, 44]

Per l'analisi del segnale elettromiografico sono applicabili l'approccio statistico e l'approccio a reti neurali, in quanto si è scelto di utilizzare lo spazio delle caratteristiche come rappresentazione del segnale EMG. Gli altri approcci oltre ad usare una rappresentazione diversa del segnale, risolvono dei problemi di natura diversa. Studi dimostrano che le reti neurali in ambiente bioingegneristico, in particolare per l'analisi EMG, hanno prestazioni significativamente migliori rispetto ad altri algoritmi, con un'accuratezza attorno al 96.40% [49].

2.3.2 Classificatori Lineari e Non Lineari

In generale i classificatori che utilizzano segnale EMG si basano sulla probabilità che un determinato segmento di segnale appartenga o meno a una classe di movimento, quindi a una densità o a una funzione di probabilità.

É possibile equiparare un classificatore a un insieme di funzioni discriminatorie lineari. In questo caso si parla appunto di classificatori lineari. Questi hanno il vantaggio di essere semplici e di facile computazione.

Tuttavia in problemi in cui non sempre è possibile separare linearmente le classi (come nel caso di segnale EMG) è spesso maggiormente indicato l'utilizzo di un classificatore non lineare, perché presenta prestazioni migliori.

Di seguito verranno descritti brevemente i due tipi di classificatore, fornendo degli esempi concreti che verranno poi impiegati in questo studio di tesi.

Classificatori lineari

I classificatori lineari cercano di separare le osservazioni attraverso funzioni di discriminazione nello spazio delle caratteristiche. Le funzioni di discriminazione lineari creano un iperpiano nello spazio vettoriale delle caratteristiche d-dimensionale e hanno la forma:

$$g(x) = w^t x + w_0 = 0 \tag{2.18}$$

dove $w = [w_1 w_2 w_3 \dots w_n]$ rappresenta il vettore dei pesi e w_0 rappresenta la soglia.

Se consideriamo x_1 x_2 come due punti nell'iperpiano di decisione, la relazione :

$$0 = w^T\ x_1 + w_0 = w^T\ x_2 + w_0 \rightarrow w^T\ (x_1 - x_2) = 0$$

dato che $x_1 - x_2$ è una quantità che sicuramente risiede nell'iperpiano di decisione, si dimostra che il vettore w è ortogonale all'iperpiano di decisione e ne indica la direzione. Si può inoltre definire la quantità

$$d = \frac{|g(x)|}{\sqrt{w_1^2 + w_2^2}}$$

dove $g(x)$ è una misura della distanza dei punti dall'iperpiano di decisione e prende valore positivo da un lato dell'iperpiano e valore negativo dall'altro lato: questo è utile per definire a quale classe appartiene il generico punto x. Per avere un output di classificazione spesso si utilizza anche $y = sign(g(x))$, utile per paragonare due classi.

Come si vede in Figura 2.9 il confine di decisione viene definito da una funzione lineare con funzione di appartenenza $\{x \mid wx + b = 0\}$ e funzione di classificazione $f(x) = sign(wx + b)$.

In generale i classificatori lineari sono più robusti al rumore e a derive del segnale. Questo è dovuto ai maggiori vincoli imposti ad un classificatore lineare (meno parametri liberi da addestrare) e implicitamente resistenti all'overfitting [50]. Comunque con la presenza di outlier e di forte rumore perfino i classificatori lineari hanno un degrado delle performance.

In Figura 2.10 [50] viene evidenziato come un outlier può portare al restringimento del margine di classificazione e come un campione misclassificato

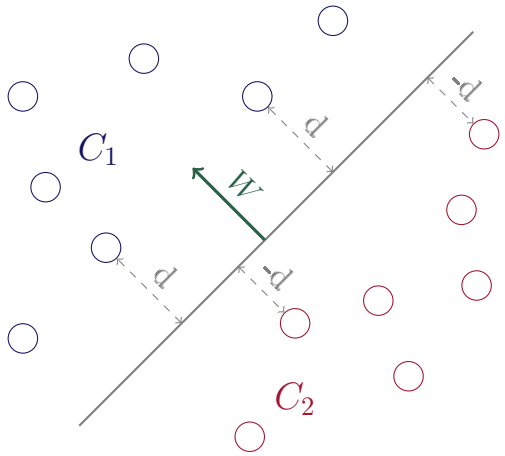


Figura 2.9: Viene mostrato l'iperpiano che divide le due classi C_1 C_2 e il vettore $\vec{w}$ che indica la direzione di decisione

a causa del rumore possa portare una falsa indicazione sulle performance del classificatore. Questi aspetti possono essere mitigati dalla regolarizzazione dei dati di training, per un classificatore più robusto al rumore e agli outliers [51].

Per fornire un esempio di classificatore lineare, di seguito verrà descritto l'analisi lineare del discriminante (LDA).

L'Analisi del Discriminante (LDA) L'Analisi del Discriminante [27] è un algoritmo che cerca di costruire una funzione discriminante tale da massimizzare la probabilità che sia riconosciuta la classe C_g avendo osservato la caratteristica $\bar{f}$, quindi creare l'associazione $< C_g, \bar{f} >$ per cui è massimizzata la probabilità $P(C_g|\bar{f})$. I classificatori vengono definiti come

$$C_g^* = argmax_(C_G)P(C_g|\bar{f}) \tag{2.19}$$

La regola di Bayes

$$P(C_g|\bar{f}) = \frac{P(\bar{f}|C_g)P(C_g)}{P(\bar{f})} \tag{2.20}$$

inserita nell'equazione 2.19 definisce l'espressione del classificatore

$$C_g^* = argmax_{c_g}\{ln\ P(\bar{f}|C_g) + ln\ P(C_g) - ln\ P(\bar{f})\} \tag{2.21}$$

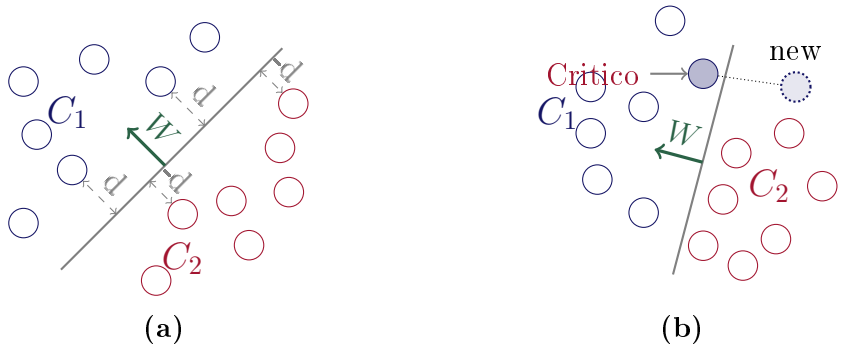


Figura 2.10: Peggioramento delle performance. **(a)** Situazione iniziale. **(b)** Peggioramento in seguito alla presenza di un outlier (*new*)

L'espressione 2.21 ci consente di trovare la classe di movimento C_g^* in funzione della probabilità del set di caratteristiche, data una classe C_g. Quest'ultima ha una distribuzione normale multivariata $P(\bar{f}|C_g) \simeq MVN(\mu_g, \Sigma_g)$, dove μ_g è il vettore media, e Σ è la matrice di covarianza della classe C_g.
Si suppone che le classi abbiano la stessa frequenza di apparizione e che la matrice di covarianza sia comune a tutte le classi. Questo ci consente di affermare che le probabilità di C_g e di f siano costanti (affermazione che può essere considerata vera solo per grandi dataset). É necessario affermare che le caratteristiche siano normalmente distribuite all'interno delle classi, per assicurare che il classificatore associato sia lineare o al più quadratico. L'assunzione che la matrice di covarianza sia condivisa consente di affermare che il classificatore risultante potrà essere lineare, in caso contrario potrebbe essere necessario un classificatore quadratico (Figura 2.11).

Da queste osservazioni il problema di classificazione viene ridotto:

$$C_g^* = argmax_{C_g}\{f^T\mu_g - \frac{1}{2}\mu_g^T\epsilon^(- 1)\mu_g\} = argmax_{C_g}\{d_{c_g}\}$$

dove $d_(c_g)$ è la funzione lineare discriminante.
Per definire questa funzione devono essere stimati i suoi parametri μ e Σ, in modo da poter posizionare le distribuzioni incognite

$$\mu_g = \frac{1}{K_g}\sum_{k=1}^{K_g} f_{c_g}k \tag{2.22}$$

$$\Sigma = \frac{1}{G}\sum_{g=1}^{G}\frac{1}{K_g - 1}(f_g - m_g)(f_g - m_g)^T \tag{2.23}$$

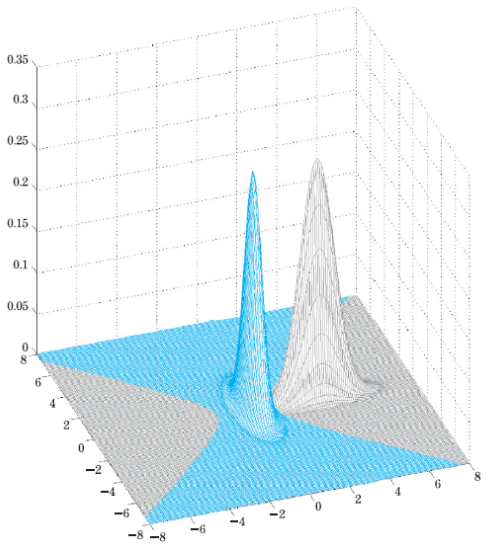


Figura 2.11: Esempio di due classi equiprobabili nello spazio bidimensionale. I vettori delle caratteristiche in entrambe le classi sono normalmente distribuiti con diverse matrici di covarianza. In questo caso la curva di decisione è un iperbole [44].

dove K_g è la cardinalità delle classi di movimento in esame. Otteniamo quindi un vettore μ_g di dimensione K_g e una matrice di covarianza che per i motivi spiegati sopra si presuppone condivisa tra le classi.

Una volta effettuata la stima dei parametri è sufficiente verificare

$$C_g^* = argmax_{c_g}\{d_{c_g}\} \tag{2.24}$$

dove $g(f) = d_{C_g}$ rappresenta la funzione di discriminazione che massimizza la funzione di appartenenza.
In letteratura sono riportati anche altri approcci al problema dell'analisi del discriminante che possono essere scelti a seconda della forma della matrice di covarianza: per una covarianza diagonale $\Sigma = \sigma^2 I$ è possibile usare una distanza euclidea per la separazione delle classi $d_\epsilon = |f - \mu_g|$, che rappresenta dei cerchi di raggio d_ϵ, ipersfere di classificazione; se invece la matrice Σ non è diagonale allora è più indicata la distanza di Mahalanobis definita come $\sqrt{(f-\mu_g)^T \Sigma^(-1)(f-\mu_g)}$, che rappresentano delle iperellissi. Essendo Σ simmetrica è possibile diagonalizzarla e ottenere la rispettiva normalizzazione sulle caratteristiche [44].

In letteratura viene riportato che l'analisi del discriminante rispetto alle reti neurali comporta una crescita quadratica dello sforzo computazionale

rispetto al numero di caratteristiche. Lo spazio dei parametri da ricercare può essere molto alto al crescere delle classi. Avendo d caratteristiche e l classi, il vettore μ_g ha cardinalità d, mentre stimare la matrice di covarianza simmetrica condivisa ha un costo $\frac{d^2}{2}$, quindi la complessità cresce quadraticamente rispetto alla dimensione del vettore di caratteristiche, mentre in una rete neurale lo spazio dei parametri da stimare cresce linearmente con l'aumentare dello stesso vettore [44].

I classificatori LDA sono molto utilizzati, nonostante non sia sempre ragionevole presupporre che in tutti i casi di utilizzo assumere Gaussiane le distribuzioni sia corretto. Il motivo principale del loro utilizzo va ricercato nella buona divisione effettuata sullo spazio delle caratteristiche dal punto di vista del classificatore. Viene in oltre riportato in studi come quello svolto da Hastie Et Al. [52] che assumere la distribuzione Gaussiana porta ad avere delle buone proprietà del classificatore e che il classificatore LDA mantiene buone performance anche dopo diverso tempo.

Classificatori non lineari

Alcune informazioni nei dati non sono estraibili attraverso classificatori lineari, perché questi tendono a implementare semplici divisioni nei gruppi di caratteristiche identificate, oppure la conoscenza che è possibile estrarre non è sufficiente ad una corretta classificazione. É per questo che sono nati i classificatori non lineari, oggi sempre più diffusi nella Pattern Recognition.

In generale la semplicità in letteratura viene preferita quando sono disponibili informazioni limitate sul fenomeno che vogliamo apprendere, o vogliamo avere maggior controllo sulle proprietà del classificatore. Tuttavia se disponiamo di sufficienti dati e informazioni sui dati di training e sul fenomeno che andiamo a studiare è preferibile una classificazione con metodi non lineari. Questo perché, con le dovute precauzioni (si veda sezioni 2.3.1 e 2.3.2), consente di estrarre maggiore informazione dai dati e quindi ridurre l'errore di classificazione rispetto ai classificatori lineari o a poter apprendere dati che sono intrinsecamente non lineari [50, 49].

Nei prossimi paragrafi verranno descritti nel dettaglio due dei principali classificatori non lineari: le reti neurali di tipo Feed Forward e le Self Organizing Feature .

Reti Neurali Feed Forward (FFNN) Il più semplice esempio di rete neurale è il percettrone che nasce dallo studio dei neuroni biologici. I neuroni accumulano cariche elettriche, la quantità di carica immagazzinata dipende sia dall'intensità del segnali provenienti dalle presinapsi, sia dalla forza di tali

collegamenti. Questo fenomeno può essere rappresentato come una somma pesata di un vettore x, dove il vettore corrisponde all'intensità del segnale presinaptico mentre i pesi w corrispondono alla forza del collegamento delle sinapsi con i neuroni presinaptici.

$$\sum wx$$

Il neurone, una volta che ha accumulato sufficiente energia, le libera sotto forma di impulso elettrico attraverso il suo assone, che andrà a stimolare gli altri neuroni a valle. Questo concetto può essere espresso in forma:

$$y \quad = f(\sum wx + w_0) f(x) \quad = sigm(x) \tag{2.25}$$

Il percettrone riesce a generare una funzione di separazione al più lineare. Prendiamo come esempio un percettrone a due ingressi $\bar{x} =< x_1, x_2 >$: supponendo la dimensionalità dell'output pari ad uno e prendendo la sua funzione di classificazione, l'output di classificazione avrà la forma:

$$y = w_1 x_1 + w_2 x_2 + w_0$$

che corrisponde ad una funzione di discriminazione lineare: una retta in un piano e, nel caso più generale, un iperpiano d-dimensionale. Non sempre questa funzione di discriminazione può essere sufficiente a classificare le classi di ingresso. In letteratura è ben noto il problema di apprendimento della funzione logica XOR, la quale non è separabile da un classificatore lineare perché non è possibile generare una retta di discriminazione nel piano $< x_1, x_2 >$ tale che le classi vengano correttamente separate.

Per ovviare a questo problema sono state introdotte reti neurali più complesse (FNN in figura 2.12); inserendo un secondo layer si ottiene come funzione di discriminazione un poliedro formato dall'intersezione degli iperpiani di decisione. Quindi il primo layer crea gli iperpiani e il secondo layer seleziona le aree separate dagli iperpiani. Questo approccio non può selezionare aree qualsiasi ma dipende dalle posizioni relative delle classi sull'ipercubo e dalla loro separabilità lineare o meno. Aggiungendo un ulteriore livello arriviamo nella condizione in cui il primo livello compone gli iperpiani, il secondo li raggruppa per creare le aree, mentre l'ultimo raggruppa queste aree in classi. Quindi un MLP di tre livelli è sufficiente a classificare qualsiasi composizione di iperpiani.

L'apprendimento delle reti neurali viene spesso associato ad un problema di minimizzazione di una funzione di costo espressa rispetto ai pesi, in modo da ottenere la matrice dei pesi che meglio approssima la correlazione

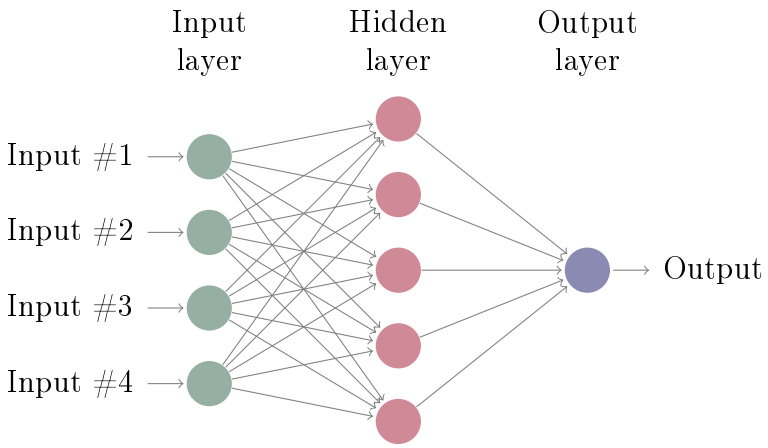


Figura 2.12: MultiLayer Perceptron a tre livelli.

tra le classi e i membri ad essa appartenenti. Per l'apprendimento si utilizza un dataset composto da coppie $< x, t >$, dove x rappresenta il vettore degli ingressi e t rappresenta il target di classificazione. Il vettore degli ingressi viene computato dalla rete neurale, il cui output viene confrontato con il target fornito: se la rete ha previsto una classe errata si procede all'aggiornamento dei pesi, che minimizza una cifra di merito, altrimenti si esegue nuovamente il procedimento con un'altra coppia di input. Così facendo si effettua il training della rete neurale.

Per le reti multi-livello viene utilizzato il metodo della back propagation, che consiste nel propagare ad ogni livello la computazione in avanti e all'indietro, in modo da poter calcolare quanto la previsione della rete differisce dall'ingresso fornito, e successivamente aggiornare i pesi per ottenere un risultato migliore alla prossima epoca. L'algoritmo quindi si divide in due fasi, derivanti dalla propagazione doppia della computazione. Mentre per il percettrone ad un livello è necessario calcolare solo la previsione della rete per poter calcolare il nuovo valore della matrice dei pesi, nel MLP viene eseguita sia la propagazione diretta e che la propagazione inversa (da cui prende il nome). Nella prima fase viene propagata la previsione del training pattern attraverso la rete neurale per produrre il valore dell'attivazione che verrà confrontato con il target desiderato; si procederà successivamente propagando il risultato inversamente alla direzione di calcolo per generare i delta associati ai neuroni nascosti.

Una volta calcolati i delta viene calcolato il gradiente dei pesi per mezzo del quale vengono calcolati i nuovi pesi, con la formula:

$$\Delta w_{ij} = -\alpha \frac{\partial E}{\partial w_{ij}}$$

dove E rappresenta l'errore commesso che può essere valutato con differenti cifre di merito (una delle più comuni è l'errore quadratico medio) e α è chiamato *learning rate* perché definisce la velocità e la qualità dell'apprendimento della rete. I nuovi pesi vengono quindi definiti come la sottrazione tra i pesi vecchi e il gradiente dell'errore attorno a quelli attuali, moltiplicato per un fattore che indica di quanto ci si deve spostare sul gradiente dell'errore di classificazione per trovare i nuovi pesi. In genere un *learning rate* più basso conferisce alla rete una migliore stabilità a discapito di una maggiore resistenza all'apprendimento.

Il problema dell'apprendimento della rete neurale può essere visto anche come un problema di ottimizzazione, consideriamo per esempio l'errore quadratico medio, definito come:

$$E = (T - Y)^2$$

Immaginando una rete con un output ad una dimensione e plottando la funzione $f(y) = E$ illustrata sopra si ottiene una parabola che ha errore pari a zero (quindi il minimo) centrato sul target T. Considerando poi la dimensionalità dell'ingresso pari a due, avremo che la funzione di errore può essere riscritta come

$$E = (T - w_1 x_1 + w_2 x_2 + w_0)^2$$

che, considerando x e t costanti per una singola coppia $< x, t >$, produce una coppa parabolica con minimo centrato sopra i pesi w_1 e w_2 che producono l'errore minimo, quindi ottimi. Per questo motivo il problema dell'apprendimento può essere visto come un problema di ottimizzazione quindi scritto nella forma

$$w = argmin_w (t - y)$$

In letteratura esistono numerose tecniche per risolvere problemi di minimizzazione di una funzione, la back propagation in particolare utilizza la discesa del gradiente che è stata formalizzata sopra.

Un MLP, di cui le FNN fanno parte, può stimare un modello di complessità arbitraria, ammesso che si possa usare un numero sufficiente di neuroni nel livello nascosto, che si disponga di un data set sufficientemente grande e si abbia a disposizione tempo sufficiente. Generalmente la funzione di errore, dipendente dal tempo di addestramento, dal numero di neuroni e dal numero

di esempi disponibili, è monotona decrescente. Questo comporta che l'errore sul training set continui a decrescere aumentando il numero di epoche di addestramento. Si può però osservare che l'errore calcolato su dei dati appartenenti alla stessa famiglia ma che non sono stati usati per l'addestramento della rete producono un errore maggiore: questo fenomeno è detto *overfitting*. L'overfitting si verifica quando la rete smette di imparare il modello di classificazione che vogliamo fargli apprendere e inizia ad imparare la specificità dei dati con cui la stiamo addestrando: la rete inizierà quindi ad avere performance scadenti con dati nuovi, perdendo quindi di generalità.

Per ovviare a questo problema si usa un criterio aggiuntivo per decidere a quale epoca fermare l'apprendimento. Una criterio valido è il controllo incrociato (*cross-validation*) che si pone come obiettivo quello di impedire che il modello implementato dalla rete neurale stia diventando troppo complesso o troppo specialistico sul training set. La cross-validation richiede di selezionare una porzione del training set per essere usato come controllo incrociato (*validation set*). Qualora le performance di classificazione sul validation set fossero peggiori di quelle del training set si dovrà porre particolare attenzione: un continuo peggioramento delle performance indica chiaramente un overfitting. La differenza tra l'errore sul validation set e sul testing set presenta delle fluttuazioni non monotone e quindi dei massimi locali; questo potrebbe portare ad interrompere l'apprendimento per un momentaneo allontanamento della funzione di errore e non per un significativo peggioramento delle performance di apprendimento. Per risolvere questo problema alcune strategie di cross-validation consentono di incrementare un contatore ogni volta che si verifica un fallimento del controllo incrociato, se il numero di fallimenti è superiore ad una soglia scelta a priori significa che il primo registrato era realmente l'inizio della divergenza delle due funzioni di errore e quindi si ricaricano i pesi calcolati a quell'epoca.

Self Organizing Feature Map (SoFM) La SOFM (o Kohonen Map) implementa un mapping ordinato di una distribuzione ad elevata dimensionalità in una griglia regolare (simmetrica) a bassa dimensionalità (di solito con dimensione pari a 2).
Converte relazioni complesse tra un grande set di dati in semplici relazioni geometriche delle loro proiezioni su uno spazio di dimensioni inferiori. Così facendo comprime informazioni preservando però le più importanti informazioni topologiche e/o metriche dei dati originali.

Come possiamo vedere dalla figura 2.13, la SOFM è costituita da uno strato di unità, interconnesse tra loro (rete neurale), alle quali viene dato in

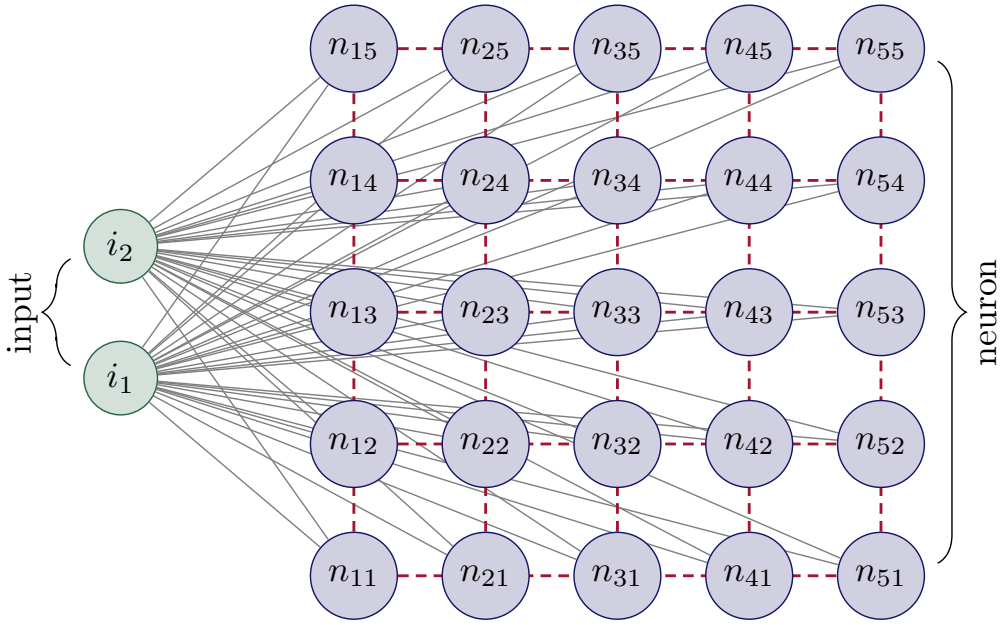


Figura 2.13: Griglia di nodi che rappresenta la Self Organizing Feature Map.

ingresso un vettore di componenti. I pesi delle connessioni tra il livello di input e quello di output vengono aggiornati grazie al processo di apprendimento che permette alla rete di auto-organizzarsi, mentre le connessioni tra i neuroni del livello di output presentano pesi che producono eccitazione tra i neuroni limitrofi e inibizioni tra i neuroni lontani.
Il contributo più innovativo al classico paradigma di apprendimento competitivo è proprio quello di aver introdotto il concetto di vicinanza (*neighborhood*), e di aver esteso l'eccitazione ad un gruppo di neuroni nelle vicinanze del neurone vincente. L'utilizzo di una *neighborhood function* permette di preservare le proprietà topologiche dello spazio di input.
Generalmente una SOFM consiste in una griglia regolare 2D di nodi. Un modello di una certa osservazione è associato ad ogni nodo. L'algoritmo calcola i modelli in modo che essi descrivano in maniera ottima il dominio delle osservazioni. Questi sono automaticamente organizzati in ordine 2D significativo, in cui i simili sono più vicini tra loro nella griglia rispetto ai più dissimili. In questo senso, la SOFM è un *similarity graph*, e anche un *clustering diagram*. Il suo calcolo è un processo parametrico di regressione ricorsiva.
Supponendo che un pattern di ingresso abbia N feature e sia rappresentato da un vettore x in uno spazio di N dimensioni, l'algoritmo di apprendimento

delle reti di Kohonen si sviluppa come segue:

1 Si inizializzano i vettori peso $w_j(0)$ sinaptici con valori casuali a partire da un set di caratteristiche disponibile, dove $j = 1, ..., l$ ed l è il numero di neuroni nel reticolo.

2 Si prende un campione x dalla distribuzione in ingresso.

3 Si individua il neurone vincente $i(x)$ utilizzando il criterio della minima distanza Euclidea $i(x) = argmin_j(|x(n) - w_j|)$.

4 Si esegue l'aggiornamento dei pesi per tutti i neuroni utilizzando la seguente formula $w_j(n+1) = w_j(n) + \eta(n)h_{j,i(x)}(n)(x(n) - w_j(n))$, dove $\eta(n) = n_0 exp(\frac{-n}{\tau^2})$ è il *learning rate* che decresce esponenzialmente all'aumentare del tempo n, e $h_{j,i(x)}(n)$ è la funzione di vicinanza centrata nel neurone vincente $i(x)$.

5 Si continua con lo step 2 fino a quando non vengono più osservati cambiamenti rilevanti nella *feature map*.

Nella procedura di Pattern Recognition, la SOFM trova il neurone vincente con la migliore similarità tra il suo vettore dei pesi e il vettore di feature in input. Quindi, le coordinate 2D del neurone vincente sono le nuove componenti del feature vector.

3. Obiettivi e scelte progettuali

> Only those who will risk going too far,
> can possibly find how far they can go
>
> Christopher McCandless

Questo lavoro di tesi si basa su uno studio precedente del Politecnico di Milano [1], nel quale si analizzavano le tecniche di classificazione del segnale elettromiografico multicanale per protesi alla spalla mediante l'utilizzo di otto canali con un classificatore LDA.

Il motivo principale per cui si scelto di utilizzare il lavoro di Rivela e Scannella è stato di poter utilizzare i dati precedentemente acquisiti e concentrarsi sulla progettazione del sistema di classificazione nella sua completezza. Così facendo si è potuto non solo espandere una trattazione esistente, completarla ed esplorare strade non trattate nella precedente, ma anche poter confrontare i risultati ottenuti, valutando quantitativamente i benefici degli approcci utilizzati.

L'obiettivo principale di questa tesi è la progettazione di un dispositivo indossabile adatto ad un utilizzo in tempo reale. Infatti il segnale sEMG oggi viene usato prettamente in ambito diagnostico e non viene particolarmente preso in considerazione per protesi ed esoscheletri, soprattutto per l'articolazione della spalla. É importante iniziare ad approfondire questo problema cercando di sviluppare un classificatore all'interno di un dispositivo indossabile, necessario per il suo utilizzo all'interno ad esempio di una protesi.

Per ottenere questo risultato si è posta particolare attenzione a dettagli pratici ed implementativi. In particolare sono stati messi in discussione alcuni aspetti dell'implementazione di Rivela e Scannella, come si vedrà nel dettaglio nei prossimi capitoli. É stato importante anche la scelta un dispositivo adeguato, tenendo conto delle dimensioni, della potenza di calcolo e dei consumi, tre caratteristiche fondamentali di un dispositivo indossabile. É stata quindi cruciale non solo la scelta delle caratteristiche e di un classificatore adeguato, in modo tale da rispettare vincoli temporali e le performance di classificazione, ma anche la scelta del linguaggio di programmazione, degli algoritmi e del dispositivo.

L'intero sistema è stato diviso in due parti, ognuna con le sue competenze specifiche. La prima parte, definita come fase di training, è implementata con *Matlab* e riprende da vicino il lavoro di Rivela e Scannella, implementando però anche una semplice interfaccia utente ed effettuando delle ottimizzazioni del codice e implementazioni di percorsi alternativi valutati durante questo studio. Questo componente carica i dataset EMG ed effettua il lavoro di classificazione offline per tutti i segmenti del segnale presente. Una volta effettuato il training del classificatore attraverso il programma Matlab il modello viene esportato per essere poi caricato dalla seconda parte, implementata sul dispositivo indossabile. Il dispositivo carica il modello contenente il classificatore e tutte le informazioni necessarie per implementare la catena di elaborazione. In questo modo può analizzare il segnale EMG in real time e comunicare la classe di movimento identificata.

Nelle fasi finali il sistema di classificazione è stato testato su un segnale elettromiografico acquisito con la scheda *Eracle*. Questo particolare test ha consentito di valutare il classificatore proposto su una scheda inseribile in una protesi a basso costo [53].

É stato esplorato ogni aspetto della progettazione e sviluppo del dispositivo: si è partiti dall'analisi del dominio analizzando con occhio quantitativo il dataset su cui il progetto si basa, considerando il protocollo di acquisizione, la strumentazione con cui è stato acquisito e i movimenti scelti. Ci si è focalizzati in particolar modo sulle caratteristiche da utilizzare, tenendo presente i requisiti temporali ai quali era necessario aderire ma cercando allo stesso modo di ottenere delle buone performance di classificazione. Il design del classificatore è stato trattato in parallelo alla scelta delle caratteristiche, in quanto estrazione delle caratteristiche e la tipologia di classificatore hanno un legame troppo stretto per essere trattati separatamente: è stato necessario comporre sei percorsi di classificazione che hanno consentito di valutare la miglior coppia di classificatori e insieme di caratteristiche. Alla luce di queste scelte si è deciso come implementare il dispositivo fisico: gli algoritmi e la qualità del classificatore sono rimasti invariati dalla versione Matlab alla ver-

sione sul dispositivo, ciò che li distingue è il linguaggio di programmazione e la ricerca di alternative software per velocizzare l'elaborazione sul dispositivo usabile.

In questo capitolo si definiscono i punti chiave che sono stati sviluppati all'interno di questo studio di tesi, spiegando nel dettaglio quali obiettivi sono stati posti. Si proseguirà successivamente a delineare l'architettura generale del sistema progettato, dal punto di vista del classificatore, della logica e della sua composizione fisica.

3.1 Obiettivi

Coerenza con il sistema esistente Il classificatore è stato progettato partendo dalla tesi di Rivela e Scannella, espandendo il lavoro precedente e utilizzando lo stesso data set per poter fornire non solo un progetto concreto per il classificatore su un dispositivo indossabile, ma anche un'analisi duale che consente di avere una visione più completa del problema di classificazione del segnale elettromiografico per amputazioni alla spalla.

Rispetto dei vincoli real time In letteratura è documentato che, per una protesi, un ritardo superiore a *300 ms* compromette l'usabilità del dispositivo. Il sistema EMG progettato si prefigge di rimanere entro questo vincolo temporale massimo, considerando anche l'overhead dovuto all'azione di controllo della protesi, il quale però non rientra negli obiettivi di questa tesi.

Device indossabili Il dispositivo su cui è stato implementato il classificatore deve avere consumi ridotti per consentire un prolungato periodo di utilizzo, deve avere sufficiente potenza di calcolo per processare il segnale e deve essere di piccole dimensioni per poter essere incluso all'interno delle protesi.

Performance statiche Il classificatore di Rivela e Scannella presenta un errore di classificazione di *19.57%* nel riconoscimento di *nove classi* di movimento[1]. Questo valore è stato preso come punto di riferimento per valutare le prestazioni di classificazione, ma l'obiettivo è almeno quello di raggiungere le performance presenti in letteratura [49] (attorno al *96.40%*), anche utilizzando un classificatore non lineare.

Performance dinamiche Si vogliono ottenere performance superiori al *96.40%*, non solo per il segnale *Steady* ma anche per la fase dinamica di *Burst*

[54]. Per poter parlare di performance dinamica occorre valutare l'errore di classificazione su un segnale reale composto di *Rest*, *Burst* e *Steady* e non solo valutare i risultati del singolo classificatore durante la fase ti apprendimento, validazione e testing.

Riduzione del numero di canali Si vuole effettuare una valutazione esaustiva dei canali del dataset di Rivela e Scannella, per definire quali canali possono essere eliminati al fine di semplificare una reale protesi. L'obiettivo è quello di diminuire il più possibile il numero di canali senza intaccare eccessivamente alle performance.

In generale, per raggiungere gli obiettivi precedentemente definiti, è stata effettuata una valutazione dell'intero percorso di Pattern Recognition, quindi dall'analisi iniziale del segnale alla classificazione da inviare poi a un controllore (in questa tesi non definito). Sono state fatte diverse scelte ponderate relative a:

- Segmentazione del segnale
- Scelta delle caratteristiche maggiormente performanti
- Valutazione di classificatori diversi, oltre a quelli trattati dalla tesi di partenza [1]
- numero di canali eliminabili

Sono stati inoltre ideati due nuovi tipi di classificatori, non presenti in letteratura, al fine di migliorare le performance real time.
Il tutto è stato implementato sia in un sistema di training che su un dispositivo indossabile.
Per effettuare un test reale è stato inoltre utilizzata una scheda di acquisizione per ottenere dei nuovi dati EMG. Tale scheda presenta però un numero di canali limitato (tre), per cui la scelta dei muscoli utilizzabili è avvenuta prendendo in considerazione lo studio sull'eliminazione dei canali svolto in questa tesi.

3.2 Architettura Sistema di Classificazione

Il sistema ha bisogno di analizzare una grande quantità di dati, un dataset composto da otto movimenti effettuati da otto soggetti diversi: per analizzare completamente i dati e addestrare il classificatore è necessario lasciare in esecuzione il programma per un tempo superiore al minuto. Queste operazioni non devono essere necessariamente eseguite ogni volta sul dispositivo

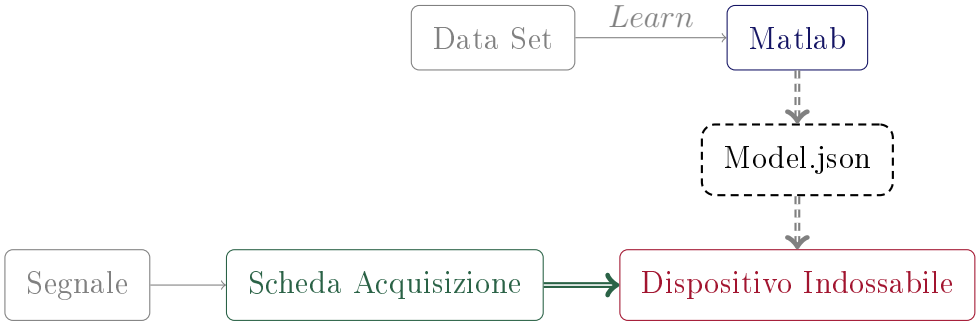


Figura 3.1: Lo schema illustra l'architettura generale del sistema di classificazione.

indossabile, per questo è stato deciso di dividere la fase di training dalla classificazione. Il modello viene costruito su un personal computer attraverso un'applicazione Matlab, la quale genera un modello contenente le informazioni necessarie a ricostruire il classificatore; successivamente questo modello può essere importato nel dispositivo per classificare il movimento. Lo schema a blocchi del sistema è rappresentato in figura 3.1.

Per testare il sistema è stata scelta una scheda di acquisizione non commerciale a tre canali. I dati di Rivela e Scannella [1] sono stati acquisiti con un *BTS FREEEMG 300* ad otto canali e una risoluzione di $1000Hz$, mentre il testing è stato fatto con una scheda *Eracle* a tre canali e frequenza di acquisizione di $270Hz$. Questa scelta particolare ci ha consentito di valutare le performance del nostro classificatore su un dispositivo low cost quindi adatto ad essere integrato in una protesi per il pubblico.

Ogni parte del sistema verrà brevemente introdotta di seguito ponendo l'attenzione al loro ruolo nell'insieme piuttosto che all'analisi di ogni singolo componente.

3.2.1 Software di Training Classificatore

La responsabilità di questo componente è di consentire il caricamento di un dataset contenente il segnale EMG, analizzarlo e produrre il modello del classificatore.

E' stato scelto Matlab come linguaggio di implementazione perché è largamente utilizzato e di veloce prototipazione, pur consentendo di mantenere alte prestazioni del codice. Inoltre è un linguaggio ampiamente utilizzato in ambito accademico e professionale, il che consentirà successive espansioni e manutenzioni. Se dovesse esserci il bisogno di produrre un codice compila-

to più performante e distribuibile sarà possibile utilizzare gli strumenti di Matlab per generare l'applicazione senza andare a riscrivere il codice.

Il programma viene gestito attraverso una maschera controllabile dall'utente dove si possono scegliere i parametri principali per il modello che si vuole generare e come output viene esportato su disco un file Json contenente il modello.

3.2.2 Dispositivo Indossabile

Il dispositivo indossabile ha la responsabilità di classificare in tempo reale il segnale EMG proveniente dalla scheda di acquisizione.

É stato scelto come dispositivo l'*Intel Edison*, il dispositivo wearable di Intel basato su un *SoC* con litografia a $22nm$ dotato di un processore *Atom* a $500Mhz$ dual core capace di eseguire due thread contemporaneamente. Incluso nel *SoC* si può trovare anche un microcontrollore a 32 bit responsabile della gestione della comunicazione e dell'hardware che è possibile collegare attraverso il socket a 70 pin. I consumi sono compresi tra $13mW$ e $35mW$ per la sola board. Il dispositivo sottoposto a profilazione da Intel ha una potenza di calcolo di $612MIPS$, leggermente inferiore ad un Raspberry PI (il quale è equipaggiato con un processore quadcore) [55].
La potenza del processore, i bassi consumi (per il tipo di architettura) e il fatto di supportare un ambiente linux sono state le ragioni che hanno portato alla scelta di questa scheda per sviluppare il classificatore.

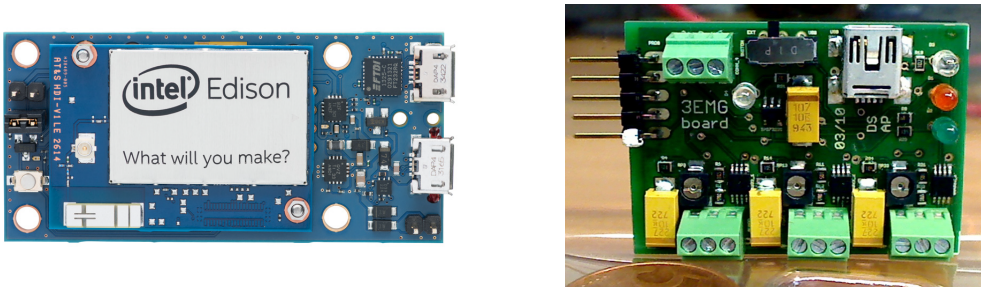
3.3 Hardware utilizzato

Vengono usate due schede hardware: la prima *Intel Edison* in figura 3.2a ospita il classificatore EMG, la seconda*Eracle* in figura 3.2b acquisisce il segnale elettromiografico.

3.3.1 Edison

La scheda Edison si compone di un *SoC* che effettua la classificazione, di una scheda di espansione che implementi le funzioni di *GpIO* e di comunicazione (*I2C* e *Seriale*). Le specifiche complete sono listate in tabella 3.1.

La caratteristica più interessante di questa scheda è di sicuro la possibilità di far eseguire un sistema Linux (la distribuzione consigliata da Intel si chiama *Yocto*) e ha consentito senza problemi di eseguire il codice Python e le librerie necessarie (*Scipy*, *Numpy*). Questa caratteristica unita alle dimensioni e ai consumi ridotti ne fanno il dispositivo ideale la prototipazione



(a) **(b)**

Figura 3.2: **(a)** Scheda Intel Edison. **(b)** Scheda Eracle.

rapida. Si nota anche la presenza di una interfaccia bluetooth 4.0 e una antenna Wi-Fi che in questo lavoro non sono state utilizzate ma per espansioni future possono essere davvero indispensabili: basti pensare alla possibilità di controllare i parametri della protesi in remoto o assistere il paziente in fase di training online.

La scheda Edison, come già citato, è equipaggiata con *Yocto*, un progetto opensource portato avanti da diversi rivenditori hardware, tra cui Intel. Lo scopo è di fornire una distribuzione Linux apposita per i dispositivi indossabili, consentendo ad ogni produttore di personalizzare la distribuzione per le sue necessità. Intel non è stata da meno fornendo una sua versione della distribuzione, che verrà usata in questa tesi.
Yocto supporta molti linguaggi di sviluppo, tra cui *Python* e *C*, che saranno i due linguaggi di programmazione utilizzati per l'implementazione del dispositivo indossabile.

3.3.2 Scheda di acquisizione Eracle

La scheda acquisisce il segnale elettromiografico da tre elettrodi di superficie alla frequenza di 270 Hz ed esegue delle elaborazioni per amplificare e filtrare il segnale prima di essere campionato dal microcontrollore ed essere reso disponibile attraverso una connessione seriale USB.

Come è visibile nello schema in figura 3.3 il percorso di elaborazione di Eracle si compone di uno stadio amplificatore *INA* che consente di fornire un'amplificazione tra i $200dB$ ai $2000dB$, selezionabile con un microswitch. Il segnale viene poi filtrato con un modulo antialiasing Sallen-Key con doppio polo alla frequenza di $150Hz$, questo stadio è particolarmente utile dato che la scheda acquisisce a $270Hz$ che non è una frequenza sufficientemente alta da

Componenti	
SoC	22 nm Intel® SoC that includes a dual-core, dual-threaded Intel® Atom™ CPU at 500 MHz and a 32-bit Intel® Quark™ microcontroller at 100 MHz
RAM	1 GB LPDDR3 POP memory (2 channel 32bits @ 800MT/sec)
Flash Storage	4 GB eMMC (v4.51 spec)
Interfacce Esterne	
SD CARD	1 Interfaccia
UART	2 Controller
I2C	2 Controller
SPI	1 Controller
USB 2.0	1 OTG controller
PWM out	32 kHz, 19,2 MHz
Power	
Input	3.3 - 4.5 V
Output	100 ma3.3V - 100ma1.8V
Consumi min	13 mW
Consumi max	40 mW
Software	
CPU OS	Yocto Linux* v1.6

Tabella 3.1: Tabella dati Edison.

evitare fenomeni di aliasing. Come ultimo stadio di elaborazione si trova un microcontrollore *PIC16F688* che effettua la conversione analogico/digitale e rende disponibile i tre canali su una connessione seriale a 57600 *bound/s*, la conversione da *RS232* a *USB* viene gestita dal cip *FT232RL*.

La scheda può essere connessa ad un PC o ad un altro dispositivo con una connessione USB di tipo host. La scheda emette due possibili stringhe: `D: 429 507 987` indica l'acquisizione dei tre campioni uno per ogni canale, ogni 100 campioni si ripete la stringa `I:a b c` ad indicare che sono stati acquisiti 100 campioni.

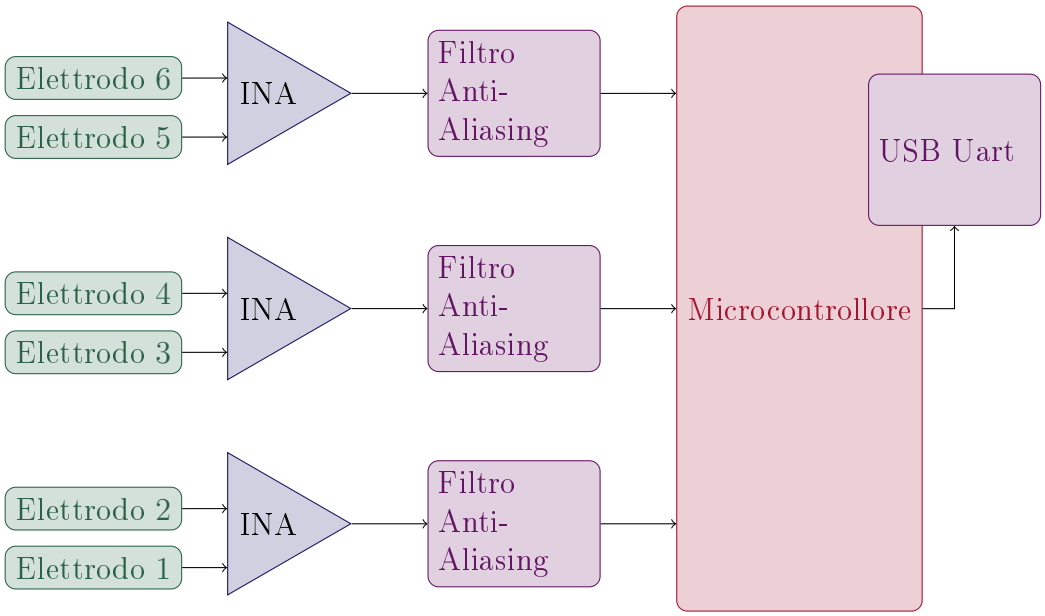


Figura 3.3: Architettura della scheda di acquisizione *Eracle*, dove sono visibili i componenti principali che la compongono.

4. Analisi del Dataset

> A mind needs books as a sword needs a whetstone if it is to keep its edge.
>
> Tyrion Lannister

Questo studio di tesi si basa sull'analisi del segnale EMG, per cui è necessario avere la possibilità di prelevare il segnale mediante strumentazione fisica o di accedere a un dataset pre-esistente e conosciuto.
Avendo a disposizione un insieme di segnali registrati presso il laboratorio di Tecnologie Biomediche MBMC all'interno del Politecnico di Milano, si è scelto di utilizzare il dataset fornitoci dalla tesi di Rivela e Scannella [1].
Questo insieme di dati infatti è stato acquisito utilizzando una strumentazione precisa e affidabile, presentando la registrazione di ben otto canali mediante elettrodi superficiali.

In un secondo momento si è riusciti a utilizzare una scheda di acquisizione Eracle prodotta e fornita dal PhD Paolo Belluco all'interno dell'azienda B10nix. Tale scheda era già stata utilizzata in tesi precedenti [56] [13].
Il limite principale di questo nuovo dataset è che utilizza unicamente tre canali, a differenza degli otto di quello precedente, per cui la discriminazione dei nove movimenti è altamente limitata. Per questo è stato utilizzato per il testing finale, con lo scopo di valutare l'accuratezza del nuovo classificatore realizzato (di cui si parlerà nella sezione dedicata) tenendo conto dei limiti della scheda.

Di seguito verrà dapprima descritto il dataset utilizzato nella tesi di Rivela e Scannella, poi verrà esposto uno studio sulla valutazione degli otto canali

da loro usati, al fine di individuare quelli eliminabili. Infine sarà esposto il dataset acquisito mediante la scheda Eracle.

4.1 Dataset di Rivela - Scannella

Il dataset presente nella tesi di Rivela e Scannella [1] è stato prelevato presso il laboratorio di Tecnologie Biomediche MBMC (Moviment Biomechanics and Motor Control) al Politecnico di Milano nell'anno accademico 2012/2013.
Vi hanno partecipato 8 soggetti sani, di cui 4 femmine e 4 maschi, con un'età media di 25 anni.

Ciascun soggetto ha effettuato 10 ripetizioni per ognuno degli 8 movimenti presi in esame. Ogni ripetizione si è svolta eseguendo le seguenti fasi:

1. raggiungimento dell'angolo target dalla posizione di Rest (riposo) in 2 secondi;
2. mantenimento della posizione target per 3 secondi;
3. ritorno alla posizione di riposo in 2 secondi.

La rilevazione è avvenuta facendo tenere al soggetto una posizione *ortostatica*, con piedi paralleli e palmi rivolti verso il corpo. Questo per limitare i muscoli usati nella fase di *Rest*.
Inoltre, per non far affaticare i muscoli della persona, i soggetti hanno deciso autonomamente l'intervallo di riposo tra una serie di movimenti e un'altra.

I movimenti in analisi, effettuati a braccio teso, sono i seguenti:

1. quattro movimenti di flesso/estensione nel piano sagittale (45°, 90°, 110°, −30°)
2. due movimenti di ab/adduzione nel piano frontale (45°, 90°)
3. due movimenti nel piano inclinato (45°, 90°)

Per costruire il dataset sono stati prelevati due segnali diversi, uno per effettuare l'analisi cinematica del movimento e uno di tipo EMG.
Per rilevare il primo tipo di segnale è stato utilizzato un sistema optoelettronico SMART-e che, mediante l'ausilio di telecamere a infrarosso e marcatori ottici di forma sferica, permette di ricostruire il sistema in 3D e individuare l'istante di inizio di ciascuna delle tre fasi precedentemente descritte e valutando la variazione dell'angolo della spalla. La frequenza di campionamento è stata di 120 Hz.

Per quanto riguarda il segnale EMG, è stato utilizzato un elettromiografo BST FREEEMG 300, che utilizza sonde EMG wireless, e la frequenza di campionamento è stata di 1000 Hz. Sono stati utilizzati otto canali corrispondenti a otto elettrodi posizionati su alcuni muscoli coinvolti nell'articolazione della spalla.
Per questo tipo di segnale è importante valutare il numero di elettrodi utilizzati (che dipende anche dal numero di classi di movimento in esame) e soprattutto il posizionamento, che può determinare l'entità del rumore e la qualità del segnale.

In totale il dataset comprende 64 segnali EMG, acquisiti dagli otto soggetti volontari per ognuno degli otto movimenti in esame (la fase di Rest è presente in ognuno dei segnali). Inoltre, per ciascuno di questi segnali, sono presenti altrettanti che ne descrivono la cinematica e, in particolare la variazione dell'angolo prodotto tra la fase di Rest e la posizione del movimento in esame.
Combinando i due vengono definite le fasi di inizio e fine steady.

4.1.1 Scelta dei muscoli e posizionamento degli elettrodi

La scelta dei muscoli su cui posizionare gli elettrodi è stata effettuata valutando 3 requisiti:

1 utilizzo del muscolo nei movimenti richiesti

2 presenza del muscolo anche a seguito di un'amputazione

3 posizionamento superficiale

Sulla base di questi punti sono stati scelti quelli riportati in tabella 4.1

Per quanto concerne il posizionamento degli elettrodi, non esistono in letteratura degli standard a riguardo, ma nella tesi viene preso in considerazione un recente atlante pubblicato da Barbero M., Merletti R. e Rainoldi A. nel 2012 [57], in cui vengono descritti i corretti posizionamenti degli elettrodi per la rilevazione dell'sEMG.
Nella tabella 4.2 si trovano le posizioni scelte per la creazione del dataset, visibili anche nella figura 4.1.

4.2 Valutazione degli otto canali

Il dataset di Rivela e Scannella precedentemente definito presenta dei segnali composti da otto canali, ognuno dei quali corrisponde al posizionamento di un elettrodo.

Canale	**Muscolo**	**Atto motorio che lo coinvolge**
1	Grande Pettorale Clavicolare	Flessione anteriore gleno-omerale
2	Grande Pettorale Sternale	Estensione all'indietro gleno-omerale Adduzione gleno-omerale
3	Serrato Anteriore	Depressione scapolare Rotazione interna della scapola
4	Trapezio Discendente	Elevazione scapolare Depressione scapolare Rotazione interna della scapola
5	Trapezio Trasverso	Depressione scapolare
6	Trapezio Ascendente	Depressione scapolare Rotazione interna della scapola
7	Infraspinato	Rotazione esterna gleno-omerale
8	Gran Dorsale	Estensione all'indietro gleno-omerale Adduzione gleno-omerale

Tabella 4.1: Muscoli scelti per l'acquisizione [1].

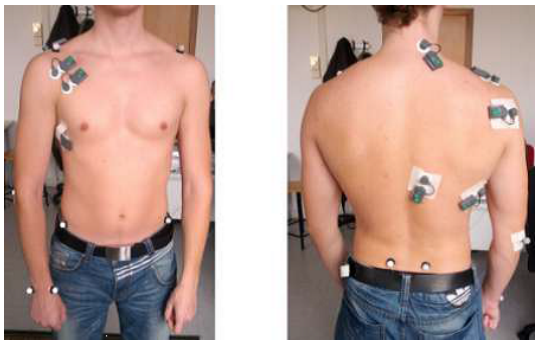


Figura 4.1: Posizionamento elettrodi sul soggetto [1].

Un problema delle protesi, a maggior ragione per quelle mioelettriche, consiste nel peso: queste devono cercare di essere il più simile possibile a un

Muscolo	Starting Point ALF	Ending Point ALF	Range consentito (%)	Posizione scelta
Grande Pettorale	Angolo Acromiale	Processo Xifoideo	0 – 45 76 - 100	25% Clavicolare 40% Sternale
Serrato Anteriore	Non è possibile individuare una ALF		Tra V e VII Costa	VII Costa
Trapezio Discendente	Clavicola Distale	VI Vertebra Cervicale	0 – 44 72 - 100	80%
Trapezio Trasverso	VII Vertebra Cervicale	Angolo Acromiale	0 – 35 70 - 100	70%
Trapezio Ascendente	Spina Scapolare	XII Vertebra Toracica	0 – 22 52 - 100	60%
Infraspinato	Punto centrale del bordo mediale della scapola	Tubercolo Maggiore	40 - 100	50%
Gran Dorsale	Punto centrale tra Tubercolo Maggiore e V Vertebra Lombare	Spina Iliaca Superiore Posteriore	0 – 16 55 - 100	0%

Tabella 4.2: Scelta del posizionamento degli elettrodi per i muscoli scelti [1].

arto reale, soprattutto visto che potrebbero recare squilibrio all'utilizzatore o problemi di postura, portando all'inutilizzo della protesi perché rigettata dal paziente.
Per alleggerirla bisogna cercare di ridurre il più possibile l'hardware utilizzato, per cui una riduzione dei canali può migliorare questo aspetto.

Inoltre così facendo si ridurrebbe la complessità per l'elaborazione del segnale, che comporta un miglioramento anche in fatto di tempi di risposta.

É stato quindi effettuato uno studio sui canali presenti nei tracciati EMG della tesi di Rivela e Scannella, cercando di valutarne le performance e di definire quelli eventualmente eliminabili.

Il grafico in figura 4.2 mostra per ogni canale la media quadratica tra tutti i soggetti in esame (otto) dei valori dei singoli movimenti. É stato scelto questo tipo di misura per eliminare i segni dovuti all'andamento del

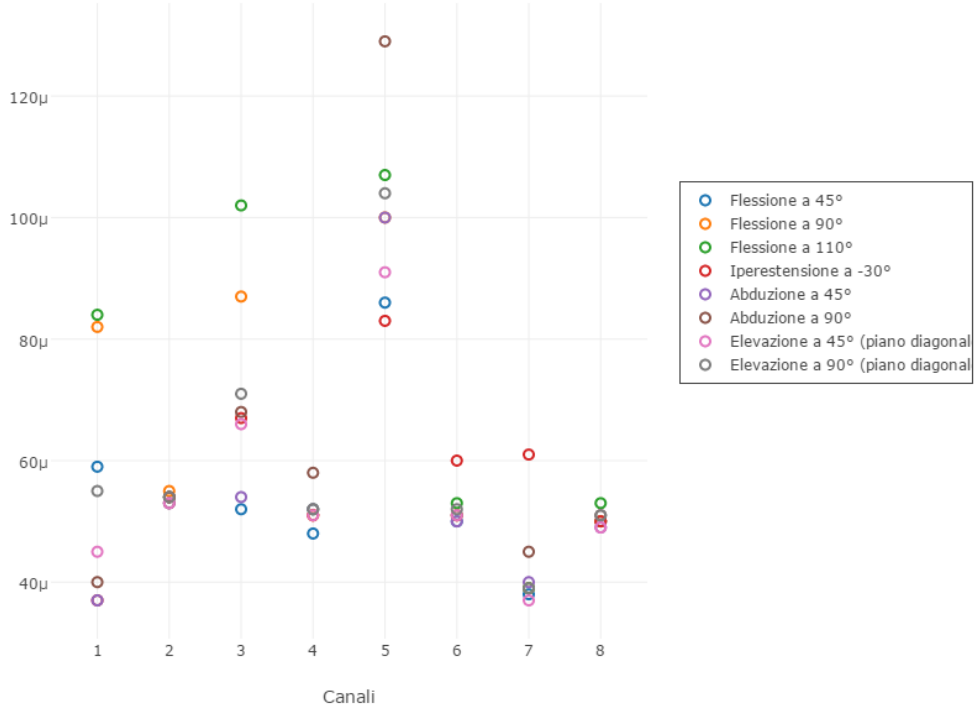


Figura 4.2: Media quadratica di tutti i movimenti per ogni singolo canale. Sull'asse y sono presenti i valori in V, mentre sull'asse x sono rappresentati i canali

segnale EMG.
La bontà o meno di un canale dipende da quanto sono distanziati i punti in verticale: maggiore sarà la distanza, maggiormente il canale sarà in grado di discriminare i diversi movimenti. Di conseguenza è possibile notare come in particolar modo i canali 2 e 8 hanno una maggiore sovrapposizione di medie dei diversi movimenti, il che li rende peggiori rispetto agli altri.
I canali che invece hanno maggiore qualità sono il numero 1 e il numero 5 (l'elettrodo di ognuno degli otto canali è stato posizionato secondo la tabella 4.1 nella sezione del dataset di Rivela e Scannella 4.1).

La figura 4.3 invece mostra la media dei valori di media quadratica dei movimenti per ogni singolo canale e il relativo scarto quadratico medio. Con questo grafico è possibile visionare la stima della variabilità dei valori assunti dai dati in input. L'esito è simile al precedente e il canale 2 e il canale 8 rimangono sempre quelli con una valutazione peggiore.

Nella figura 4.4 invece è possibile osservare la media quadratica e lo scarto

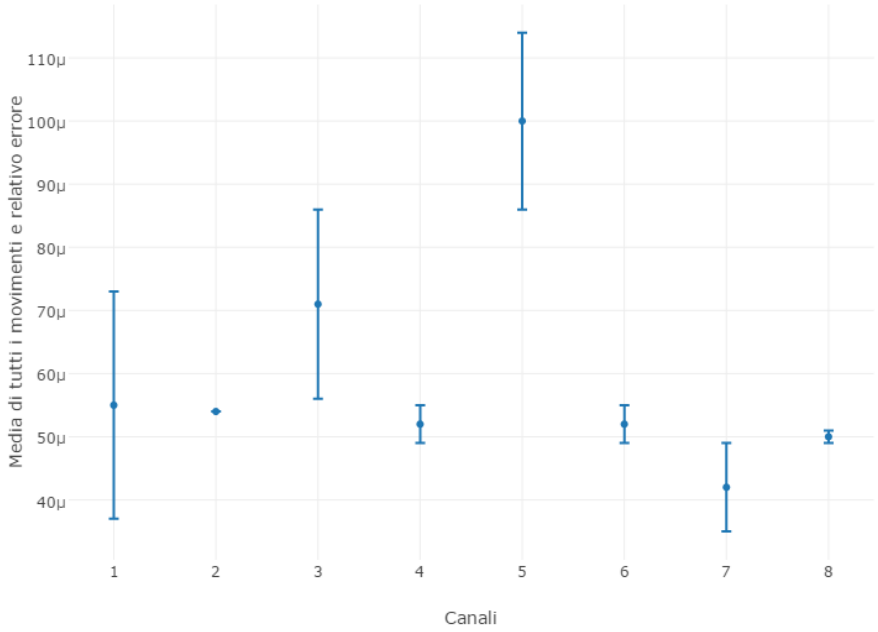


Figura 4.3: Intervallo di ampiezze assunte in media per i movimenti per ogni singolo canale. Sull'asse y i valori sono espressi in V.

quadratico medio per ogni movimento in ogni singolo canale.
In questo caso si vuole indicare la variabilità dei valori assunti per ogni singolo movimento. Verranno quindi premiati quei canali che hanno uno scarto quadratico medio inferiore, perché significa che, per il singolo movimento, rileveranno un valore più preciso a differenza di soggetto.
Osservando attentamente si può notare che i canali migliori sono l'1, il 2, il 6 e il 7. Il numero 2 però per i movimenti 5,6,7 ha un valore medio pressoché identico, il che significa che non è in grado di discriminare i tre movimenti.

4.3 Dataset prelevato con il sistema Eracle

Nel corso del progetto si è presentata la possibilità di prelevare nuovamente un segnale sEMG.
Si è scelto di effettuare una nuova acquisizione perché il nuovo dataset dà

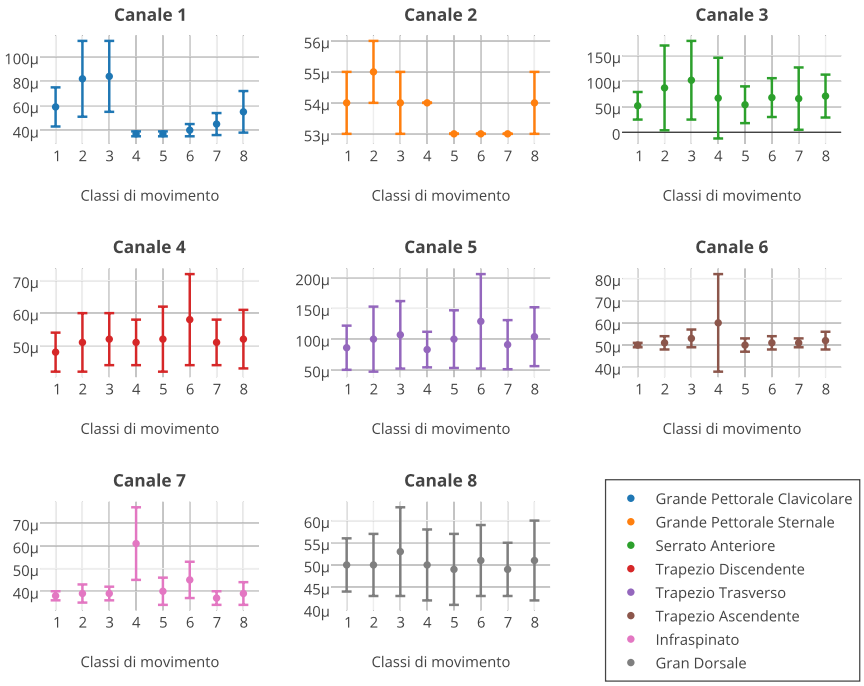


Figura 4.4: Ogni grafico contiene il range di ampiezze (espressi in V) assunte in media dal relativo movimento per uno degli otto canali.

la possibilità di valutare l'accuratezza di classificazione in una situazione fortemente svantaggiata, discriminando otto movimenti diversi(esclusa la fase di rest) con solo tre canali a disposizione. Utilizzando il nuovo segnale, acquisito dalla board Eracle (descritto al capitolo *Obiettivi e scelte progettuali* nella sezione dedicata 3.3.2), si riesce quindi a valutare meglio la robustezza del classificatore.

4.3.1 Modalità di acquisizione

Il sistema Eracle è stato ideato per il riconoscimento dei movimenti della mano.
Presenta quindi tre canali per l'acquisizione e un elettrodo di riferimento che sono fortemente limitanti per un utilizzo in una protesi alla spalla che vuole riconoscere nove classi di movimento (otto più la fase di Rest), in cui i muscoli sono meno superficiali e hanno un maggiore cross-talking.

Nonostante questo, può essere utile usare un segnale diverso per valutare la capacità del classificatore di discriminare classi diverse e simulare un utilizzo real-time.
Di conseguenza è stato necessario determinare un posizionamento degli elettrodi per avere un segnale con maggiore informazione possibile.

Posizionamento degli elettrodi

Nella tesi di Rivela e Scannella, riportata nella sezione precedente, è presente una descrizione dettagliata dei muscoli coinvolti nel movimento della spalla. La difficoltà è quella di ridurre il numero degli elettrodi da otto a tre cercando di mantenere più informazione possibile (nella tabella 4.1 nella sezione 4.1 è presente la relazione numero elettrodo-muscolo utilizzato).
Per ridurre il numero di canali è stato utilizzato lo studio precedentemente esposto.

Osservando poi la tabella che descrive anche il coinvolgimento del movimento, sono stati selezionati i seguenti muscoli, mediante una valutazione incrociata delle precedenti indagini:

- Grande Pettorale Clavicolare (canale 1 in tabella 4.1)
- Serrato Anteriore (canale 3 in tabella 4.1)
- Trapezio Ascendente (canale 6 in in tabella 4.1)

Per il posizionamento è stato seguito quello proposto dall'atlante [57].

Acquisizione

L'acquisizione è stata fatta su un singolo soggetto femminile di anni 26, senza alcun problema muscolare a livello di articolazione di spalla.
Il protocollo di acquisizione corrisponde, ad eccezione dei tempi, del numero di ripetizioni e delle modalità di rilevazione della cinematica, a quello utilizzato da Rivela e Scannella [1]. La posizione utilizzata durante la rilevazione è stata quella ortostatica. Per l'indagine cinematica, è stato usato un dispositivo Kinect in grado di rilevare i giunti del soggetto, al fine di poter calcolare l'angolo tra il busto e l'arto e differenziare la fase di burst da quella di steady. Il soggetto ha effettuato sette ripetizioni per ognuno degli otto movimenti in analisi.

L'iter di acquisizione è avvenuta nel seguente modo:

- Elevazione dell'arto fino alla posizione target in 2 secondi.
- Mantenimento della posizione per 4 secondi.

- Ritorno alla posizione di Rest in 2 secondi.
- Pausa in posizione di risposo per 2 secondi.

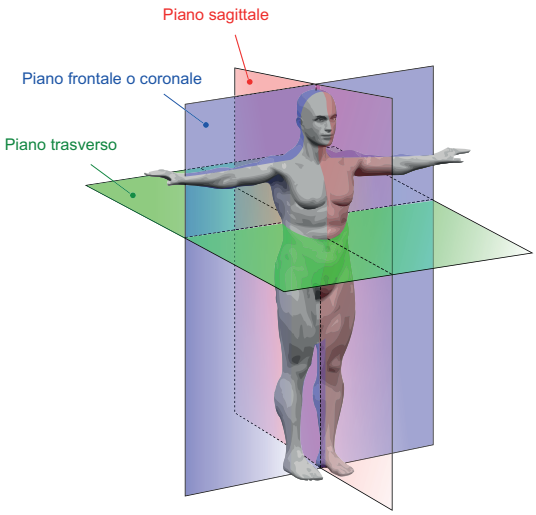


Figura 4.5: Piani di divisione del corpo per i movimenti [58].

Ciò è avvenuto per ognuno degli otto movimenti descritti nella tesi di partenza (piani citati visibili in figura 4.5):

1 Flesso/estensione a 45° sul piano sagittale

2 Flesso/estensione a 90° sul piano sagittale

3 Flesso/estensione a 110° sul piano sagittale

4 Flesso/estensione a $-30°$ sul piano sagittale (iperestensione)

5 Ab/Adduzione a 45° sul piano frontale

6 Ab/Adduzione a 90° sul piano frontale

7 Movimento a 45° nel piano inclinato[1]

8 Movimento a 90° nel piano inclinato[1].

[1]Per piano inclinato si intende il piano frontale inclinato di 45° verso il piano sagittale

Sicuramente questi movimenti non sono sufficienti per avere una completa libertà di movimento dell'articolazione della spalla.
La loro scelta è stata presa dal lavoro di Rivela e Scannella [1] e vuole essere un inizio per lo studio di questa parte del corpo, che in ambito sEMG è ancora poco presa in considerazione e necessita di ulteriori sviluppi.

5. Progettazione del sistema e fase di training con Matlab

> The saddest aspect of life right now is that science gathers knowledge faster than society gathers wisdom.
>
> I. Asimov

Lo scopo del *machine learning con apprendimento supervisionato* è quello di definire una funzione che riesca a identificare pattern tra i dati in ingresso e l'output noto (le classi di movimento), in modo tale che, a fronte di un nuovo dato, sia possibile predire la classe di movimento. Queste coppie input-output, usate per definire la funzione, compongono i *dati di training*, e la fase che permette di identificare i pattern è, appunto, la *fase di apprendimento*.

In questo capitolo inizialmente verrà descritto brevemente il metodo presente nella tesi di Rivela e Scannella[1], che è stato la base di questo lavoro. Verranno poi mostrate le diverse scelte di design del sistema e le relative valutazioni in fatto di performance, al fine di ottenere la configurazione migliore sia per quanto riguarda l'accuratezza, sia per il tempo di esecuzione, in modo tale da poter rientrare nella finestra real-time di $300ms$. Lo studio è stato svolto per ogni fase di Pattern Recognition, per cercare di ottimizzare le tempistiche totali della classificazione del movimento, dall'acquisizione alla classificazione.
Nonostante la fase di training non debba essere necessariamente così veloce, queste valutazioni sono state necessarie per poter poi implementare l'appli-

cazione sul dispositivo reale (descritto nel capitolo *Implementazione su Dispositivo Indossabile*), il quale deve avere un tempo di risposta il più breve possibile.

In seguito verrà descritta la vera e propria fase di training: partendo dalla tesi di Rivela e Scannella [1], sono state effettuate ottimizzazioni al codice presente e testati nuovi percorsi. É stata implementata anche una semplice interfaccia grafica, con la possibilità di scegliere tra i principali percorsi trattati, in modo tale da far visualizzare l'errore di classificazione dell'iter scelto anche ad un utente meno esperto.
Tutti i contenuti di questo capitolo derivano da codice Matlab, realizzato mediante la versione *Matlab R2015a*. In appendice A è presente il collegamento al repository del codice.

5.1 Descrizione del metodo presente

Il lavoro di tesi da cui si è partiti aveva l'obiettivo di valutare la miglior tecnica di elaborazione del segnale sEMG per poter attuare il controllo di una protesi mioelettrica per la disarticolazione della spalla.
Le autrici hanno valutato le diverse fasi della *Pattern Recognition*, in particolare la segmentazione del segnale, la scelta delle caratteristiche e il classificatore, cercando di ottenere la migliore classificazione possibile tra quelle presenti in letteratura, mediante l'utilizzo della fase steady del tracciato EMG. Tale segnale è stato definito mediante l'ausilio del sistema optoelettronico SMART-e, come precedentemente descritto al capitolo corrispondente *Analisi del Dataset*.

Sono state indagate sei tipologie di segmentazione diverse, differenziate per lunghezza di finestra e per incremento:

1. Lunghezza = 500 ms; Incremento = 250 ms;
2. Lunghezza = 500 ms; Incremento = 125 ms;
3. Lunghezza = 500 ms; Incremento = 62 ms;
4. Lunghezza = 250 ms; Incremento = 250 ms;
5. Lunghezza = 250 ms; Incremento = 125 ms;
6. Lunghezza = 250 ms; Incremento = 62 ms;

Per ogni tipologia sono stati applicati quattro procedimenti diversi per l'elaborazione del dataset, che vengono descritti nella tabella 5.1.

Percorso	Estrazione delle caratteristiche	Riduzione delle caratteristiche	Classificazione
1	MAV WL ZC SSC (0.3)	PCA	LDA
2	SampEn CC RMS WL	PCA	LDA
3	ZC (2° liv. con Db7) WAMP (2° liv. con Db7) (0.01) MAV (2° liv. con Db7) MYOP (4° liv. con Db5) (0.5)	PCA	LDA
4	WPT con Sym5	LDB + PCA + SOFM	LDA

Tabella 5.1: Percorsi di elaborazione nella tesi di Rivela e Scannella [1].

In ognuno dei quattro procedimenti sono state utilizzate quattro, cinque e nove classi di movimento, per poter valutare l'accuratezza di classificazione all'aumentare dei movimenti in esame.

Osservando gli errori di classificazione in fase di test, è stata definita come segmentazione migliore la combinazione 500ms-62ms, e come percorso migliore in termini di accuratezza il numero 2 (tabella 5.1).

5.2 Progettazione del sistema e valutazione delle prestazioni

A partire dal precedente codice Matlab, è stata fatta una valutazione delle performance con l'utilizzo del dataset di Rivela e Scannella.
Il requisito di real-time per una protesi è quello di 300 ms, ed è quindi importante, per poter rendere un sistema di classificazione attuabile nella realtà,

riuscire a limitare il più possibile il tempo di elaborazione senza intaccare eccessivamente l'accuratezza.
Infatti bisogna prendere in considerazione anche le fasi successive alla classificazione, come il controllo della protesi (che in questo studio non viene trattato nello specifico), perché il vincolo di real-time vale per l'intero percorso di protesi, dall'acquisizione del segnale EMG al movimento vero e proprio della protesi.

Di seguito verranno valutate le performance dei diversi percorsi eseguiti per ogni fase della classificazione.

5.2.1 Insiemi di classi

Sono stati utilizzati tre insiemi di classi di movimento di diverse dimensioni, valutando gruppi di nove, cinque e quattro movimenti [1], così composti (R=Rest, Fl=Flessione, I=Iperestensione, Ab=Abduzione, El diag=Elevazione nel piano diagonale):

Nove Classi : R, Fl 45°, Fl 90°, Fl 110°, I-30°, Ab 45°, Ab 90°, El 45°diag , El 90°diag .

Cinque Classi : R, Fl 90°, I-30°, Ab 90°, El 90°diag.

Quattro Classi : R, Fl 90°, I-30°, Ab 90°.

Su queste classi Rivela e Scannella avevano fatto le seguenti osservazioni:

- La classe di flessione a 45° viene stimata erroneamente come elevazione a 45° nel piano diagonale (11.54% delle volte), flessione a 90° (10.47% dei casi), flessione a 110° (7.29%) ed elevazione a 90° nel piano diagonale (6.02%).
- La classe di elevazione a 45° nel piano diagonale viene confusa con l'abduzione a 45° (14.24% dei casi), con la flessione a 45° (10.00% dei casi) e con l'elevazione a 90° nel piano diagonale (9.36% dei casi).
- La classe di elevazione a 90° nel piano diagonale viene confusa con l'abduzione a 90° (11.37% dei casi), con la flessione a 110° (7.78% dei casi), con la flessione a 45° (5.43% dei casi) e con l'elevazione a 45° nel piano diagonale (3.93% dei casi).
- La classe di abduzione a 90° viene confusa con l'elevazione a 90° nel piano diagonale (9.60% delle volte), con l'abduzione a 45° (5.81% delle volte) ed elevazione a 45° nel piano diagonale (2.71% delle volte).

- La classe di abduzione a 45° viene confusa con l'abduzione a 90° (4.64% delle volte), con l'elevazione a 45° nel piano diagonale (4.44% delle volte) e iperestensione a −30° (3.64%).

Questi tre insiemi di classi sono stati utilizzati per una valutazione dei percorsi inizialmente scelti, per capire la variazione delle performance in relazione alla quantità di movimenti da classificare.

5.2.2 Segmentazione del segnale

Da numerosi studi emerge che una componente fondamentale per la costruzione di un buon sistema di classificazione è l'apertura della finestra di acquisizione. Le finestre che sono state verificate sono di lunghezza *500*, *250*, *125* ms. Da precedenti lavori è evidente che una finestra con maggiore ampiezza porta a migliori performance di classificazione, includendo più informazione, e, come si vedrà nella sezione *Implementazione su Dispositivo Indossabile*, una finestra maggiore aiuta a ridurre gli errori di classificazione nella fase di transizione tra la fase di rest e la fase di steady. Si è scelto di decidere anche diverse lunghezze di offset per lo spostamento delle finestre, ma è risultato un parametro decisamente meno influente sulle performance del classificatore. Per questo motivo lo spostamento delle finestre di segmentazione verrà deciso solo in merito alle performance temporali e alle capacità computazionali del dispositivo fisico.

Sono state provate tutte le combinazioni tra lunghezza di finestra di 500, 250 e 125 e offset di 250, 125, 62. I risultati, in relazione ai percorsi elaborati, sono riportati più avanti (Tabelle 5.3, 5.4, 5.5, 5.6, 5.7, 5.8, 5.9, 5.10, 5.11). La lunghezza con maggiore informazione è risultata essere quella di 500 ms. Come già detto in precedenza, l'offset non incide particolarmente sull'accuratezza di classificazione, per cui si è scelta quello riportato nella tesi di Rivela e Scannella [1], pari a 62 ms.

5.2.3 Tempi di elaborazione dei set di caratteristiche

In questa fase si è deciso di prendere in considerazione i set di caratteristiche relativi ai primi tre percorsi di elaborazione, accantonando l'ultimo.
La scelta è stata dettata dal fatto che il set di Hudgins è ben conosciuto e largamente utilizzato, il set di Phinyomark con SampEn viene definito come il più robusto, mentre quello di Phinyomark utilizza le wavelet, un metodo diffuso nella valutazione del segnale elettromiografico. Abbiamo definito i tempi di elaborazione richiesti per il calcolo delle caratteristiche e abbiamo ottenuto i valori (approssimati) presenti in tabella 5.2.

Set di caratteristiche	Tempo di elaborazione per il calcolo di un segmento
Hudgins	2 ms
Phinyomark	3400 ms
Phinyomark2	75 ms

Tabella 5.2: Tempi di elaborazione per i set di caratteristiche in esame di un singolo segmento.

Nella precedente tesi il set di Phinyomark, con la sua SampEn, è stato definito come il miglior set di caratteristiche in termini di accuratezza. Purtroppo però, secondo le valutazioni dei tempi, si nota che il suo calcolo sia il più gravoso, con una media di 400 ms per il calcolo di una singola SampEn e 30 ms per i quattro coefficienti di CC (le due caratteristiche più complesse). Sono stati ricercati in letteratura nuovi algoritmi implementati e convalidati al fine di ridurre le tempistiche della SampEn, che altrimenti accrescerebbe di troppo i tempi richiesti per l'utilizzo in real-time.

Fra i diversi algoritmi analizzati, è stato scelto uno che utilizza per il calcolo della SampEn il metodo *kd-tree* (algoritmo Saenkd [1] [59] [60]). Questo metodo consiste in una struttura ad albero binario di ricerca multi-dimensionale, in cui ogni nodo indicizza un punto in k dimensioni.
Utilizzando questa struttura, si ha una netta diminuzione della complessità, con conseguente minor tempo di computazione per il calcolo della caratteristica (in media il tempo per il calcolo di una sola SampEn è di 35 ms).
Questo algoritmo è stato scelto anche perchè, a livello di accuratezza, non comporta un peggioramento significativo delle performance. Da qui in seguito, gli studi condotti con la SampEn si riferiscono a quella calcolata mediante l'algoritmo *saenkd*.

Per l'intero set di Phinyomark, con il nuovo algoritmo di SampEn, il tempo di elaborazione per un segmento diminuisce a circa 1100 ms, a differenza dei 3400 ms del codice pre-esistente. Purtroppo però non si riesce a rimanere all'interno della finestra per l'applicazione real-time.

L'elaborazione del set di caratteristiche di Hudgins è stato migliorato, calcolando non più le caratteristiche per ogni segmento, ma utilizzando il segnale segmentato nella sua interezza. Questo ha portato ad ottenere un tempo di calcolo inferiore a 1 ms.

[1] Algoritmo Saenkd: `http://www.mathworks.com/matlabcentral/fileexchange/48574-fast-sample-entropy-saenkd-dim--r--sig--/content/saenkd.m`

5.2.4 Riduzione delle caratteristiche

I set di caratteristiche calcolati sul segnale EMG hanno una dimensionalità elevata. Infatti per ogni segmento vengono calcolate 32, 56 e 32 caratteristiche (una per ogni canale), rispettivamente per i set di Hudgins, Phinyomark con SampEn (da ora chiamato Phinyomark) e Phinyomark con wavelet (indicato come Phinyomark2). Valutando l'intero segnale, la mole è evidentemente elevata. A tal proposito è stato necessario applicare una tecnica di riduzione della dimensionalità, la PCA, che è stata brevemente descritta nel capitolo *Pattern Recognition per il segnale EMG* al paragrafo corrispondente. Come si può notare dalle tabelle di fine sezione, senza l'utilizzo di PCA si ha un peggioramento fino al 15%, in termini di accuratezza di classificazione.
I valori delle componenti principali per la PCA, rilevati sperimentalmente, sono stati 17 per Hudgins, 56 per Phinyomark e 24 per Phinyomark 2. Prima di applicare la PCA è stata effettuata una normalizzazione sui dati, in modo tale da avere media pari a zero e deviazione standard uguale a 1. Questo perché il metodo delle componenti principali è sensibile al *problema dello scaling*, cioè alle variazioni delle ampiezze dei dati in ingresso, e questo tipo di normalizzazione è quello maggiormente usato per ovviare al problema, nonostante non ci sia una giustificazione teorica [61]. Tale normalizzazione viene effettuata calcolando la media μ_x e la deviazione standard σ_x dei dati e valutando la seguente espressione:

$$Norm = \frac{(x - \mu_x)}{\sigma_x}$$

5.2.5 Scelta del classificatore

Un altro punto fondamentale è la scelta del tipo di classificatore.
Questa dipende, oltre che dall'accuratezza, anche dalla complessità di implementazione in un dispositivo indossabile.
Sono stati valutati in particolare due tipologie di classificazione: LDA, che è il classificatore scelto dal lavoro di tesi di Rivela e Scannella ed è di tipo lineare, e una Neural Network (NN), che è un altro metodo largamente usato in letteratura di tipo non lineare.
In particolare, volendo utilizzare una tipologia semplice e poco complessa, si è deciso di ampliare lo studio delle NN con Feed Forward, utilizzando 20 e 50 neuroni, cercando di trovare un numero di neuroni non eccessivamente elevato ma con una buona accuratezza.

I classificatori lineari basati sulle funzioni di discriminazione lineare (LDA) consentono di mantenere buone performance di classificazione anche in presenza di elementi molto distanti dalla distribuzione o in presenza di rumore;

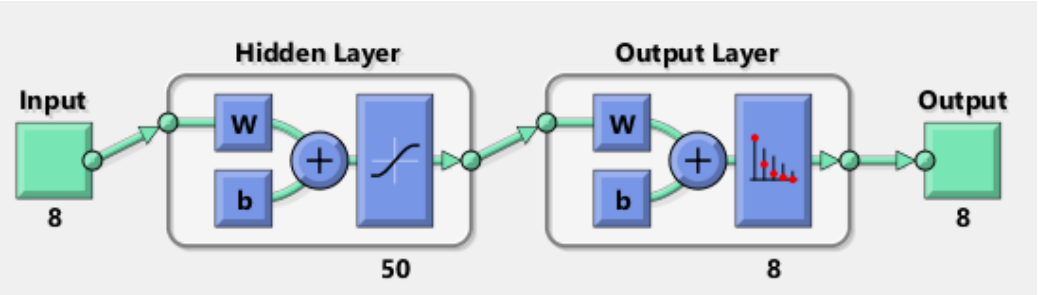


Figura 5.1: Schema della rete neurale Feed Forward utilizzato per la classificazione.

le reti neurali, invece, sono spesso utilizzate dove è necessario ottenere un classificatore veloce, che sappia adattarsi il più possibile al segnale, il che ne fanno un ottimo candidato.

LDA

Il classificatore LDA utilizzato, come già mostrato nel capitolo *Pattern Recognition per il segnale EMG* alla sezione *Classificazione*, minimizza la cifra di merito :

$$C_g = argmax_{C_g}(f^T \sum^{-1} \mu_g - \frac{1}{2}\mu_g^T \sum^{-1} \mu_g)$$

dove C_g è la classe di movimento, f^T è il vettore delle caratteristiche della finestra in esame, $\sum$ è la matrice di covarianza (assunta comune per tutte le classi) e μ è il vettore media.

Il classificatore lineare scelto associa ad ognuno degli otto gruppi di classificazione (le otto diverse classi di movimento, escludendo quella di Rest) una Normale Multivariata come funzione di densità probabilistica, stimando la covarianza dal campione analizzato. Come ogni classificatore LDA, le distribuzioni hanno distanza massima reciproca. Questo fattore aiuta a preservare buone caratteristiche di classificazione anche a distanza di tempo perché il sistema è avvantaggiato dal margine di classificazione garantito dalla probabilità sopra enunciata.

Rete Neurale di tipo Feed Forward (FFNN)

La rete neurale utilizzata è un perceptrone multilivello a tre livelli.

Come si può vedere nello schema 5.1, la rete neurale utilizza le seguenti funzioni di attivazione.

- **Layer di Input.** Viene utilizzata la funzione *MapMinMax* definita come :

$$o = y_{min} + x_{gain}(i - x_{offset}) \tag{5.1}$$

- **Hidden Layer.** Viene utilizzata la funzione *TanH* definita come:

$$o = tanh(i) * \vec{w} - w_0 \tag{5.2}$$

- **Layer.** Viene utilizzata la funzione *SoftMax* definita come:

$$o = \frac{e^{\vec{i} - max(\ \vec{i}\ )}}{\sum_{n=1}^{N} e^{i_n - max(\ \vec{i}\ )}} \tag{5.3}$$

Si è scelto questo tipo di rete neurale perché è congruente con il ben conosciuto *Neural Network tool* di Matlab. Il percorso completo è definito nel modo seguente :

$$C = SoftMax\left(\sum_{j=0}^{M} w_j\ tanh\left(\sum_{i=0}^{D} w_{ji}\,(y_{min} + x_{gain}(x_i - x_{offset}))\right)\right) \tag{5.4}$$

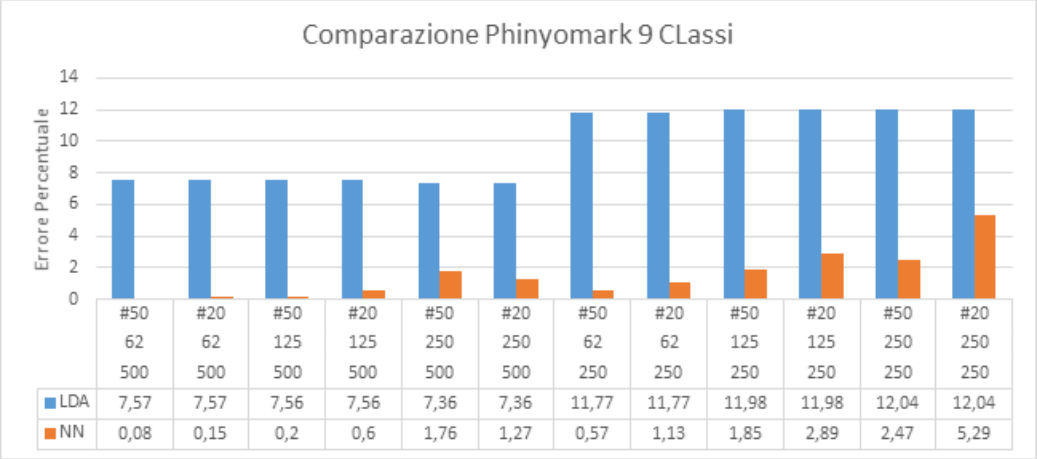
Le considerazioni fin qui ottenute e la prossima sezione sulle performance utilizzano unicamente il segnale steady, così come esposto nella tesi di Rivela e Scannella. Successivamente verrà analizzato un ulteriore percorso di elaborazione (sezione 5.3), il quale prende in considerazione anche la fase transiente del segnale.

5.2.6 Performance dei percorsi di elaborazione

Per ciascuna tipologia di segmentazione e per ognuno dei tre classificatori (di cui due FFNN) è stato valutato l'errore di testing con l'utilizzo di nove classi di movimento per i due set di caratteristiche più performanti, come è possibile vedere nelle tabelle successive 5.2 e 5.3.

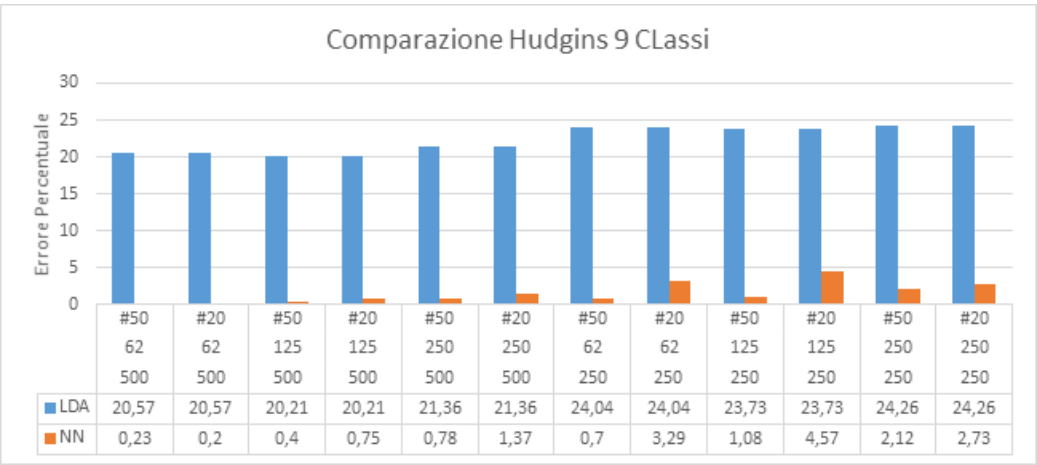
In particolare i dati presenti nel dataset sono stati suddivisi in maniera del tutto casuale. Nel caso di LDA il 60% dei dati è stato utilizzato per il training e il restante 40% per il testing, mentre nelle FFNN il 70% è stato usato per il training, il 15% per il testing e il restante 15% per la validation. Inoltre sono stati analizzati anche i tempi di classificazione, a causa del requisito real-time da rispettare.

Come si vede in figura5.2 esiste una notevole differenza in entrambi i classificatori a seconda della finestra scelta, mostrando una maggiore sensibilità



	#50 62 500	#20 62 500	#50 125 500	#20 125 500	#50 250 500	#20 250 500	#50 62 250	#20 62 250	#50 125 250	#20 125 250	#50 250 250	#20 250 250
LDA	7,57	7,57	7,56	7,56	7,36	7,36	11,77	11,77	11,98	11,98	12,04	12,04
NN	0,08	0,15	0,2	0,6	1,76	1,27	0,57	1,13	1,85	2,89	2,47	5,29

Figura 5.2: Errore di classificazione di classificatore LDA e FFNN per le finestre prese in esame per l'insieme delle caratteristiche di Phinyomark. Per ogni barra il primo valore indica il numero di neuroni, il secondo l'offset (in ms), il terzo la lunghezza di finestra (in ms).



	#50 62 500	#20 62 500	#50 125 500	#20 125 500	#50 250 500	#20 250 500	#50 62 250	#20 62 250	#50 125 250	#20 125 250	#50 250 250	#20 250 250
LDA	20,57	20,57	20,21	20,21	21,36	21,36	24,04	24,04	23,73	23,73	24,26	24,26
NN	0,23	0,2	0,4	0,75	0,78	1,37	0,7	3,29	1,08	4,57	2,12	2,73

Figura 5.3: Errore di classificazione per classificatore LDA e FFNN per le finestre prese in esame per l'insieme delle caratteristiche di Hudgins. Per ogni barra il primo valore indica il numero di neuroni, il secondo l'offset (in ms), il terzo la lunghezza di finestra (in ms).

del set di caratteristiche di *Phinyomark*. Confrontando le performance di Hudgins in figura 5.3 si nota che questo set non soffre di questa problematica, tuttavia con l'utilizzo del classificatore LDA l'errore è sempre superiore rispetto a quello calcolato con il set di Phinyomark.

Per quanto riguarda l'utilizzo del classificatore LDA sono stati confermati i risultati ottenuti nella tesi di Rivela e Scannella [1], migliorando i tempi di calcolo del feature set di Phinyomark (quello ritenuto maggiormente accurato) con l'utilizzo del nuovo algoritmo per la SampEn.
Purtroppo la classificazione rimane comunque al di sopra dei tempi di real-time richiesti in letteratura.

In entrambe le comparazioni risulta evidente che, mediante l'utilizzo delle due reti neurali, si ha un netto miglioramento nella classificazione.
In generale l'utilizzo di 20 neuroni anziché 50 provoca, anche seppur lieve, un peggioramento in fatto di accuratezza.

Nelle tabelle successive sono riportati i risultati nel dettaglio, divisi per numero di classi. Questo studio non vuole valutare il numero di classi ottimale da utilizzare, ma vuole fare un'analisi di sensitività sul numero di movimenti classificabili, mettendo in risalto il peggioramento delle performance da nove classi a cinque o a quattro; questo può fornire informazioni su quanto sia critico o no il problema e quanto i diversi classificatori siano sensibili a queste variazioni. L'obiettivo di questo studio, infatti, è quello di avere un classificatore a nove classi di movimento che mantenga buone performance anche incrementando il numero di classi.

Finestra (ms)	Offset (ms)	Errore LDA	Errore FFNN	Hidden Neurons
500	62	20.57%	0.23%	50
500	62	20.57%	0.2%	20
500	125	20.21%	0.4%	50
500	125	20.21%	0.75%	20
500	250	21.36%	0.78%	50
500	250	21.36%	1.37%	20
250	62	24.04%	0.7%	50
250	62	24.04%	3.29%	20
250	125	23.73%	1.08%	50
250	125	23.73%	4.57%	20
250	250	24.26%	2.12%	50
250	250	24.26%	2.73%	20

Tabella 5.3: Hudgins con 9 classi

Finestra (ms)	Offset (ms)	Errore LDA	Errore NN	Hidden Neurons
500	62	7.57%	0.08%	50
500	62	7.57%	0.15%	20
500	125	7.56%	0.2%	50
500	125	7.56%	0.6%	20
500	250	7.36%	1.76%	50
500	250	7.36%	1.27%	20
250	62	11.77%	0.57%	50
250	62	11.77%	1.13%	20
250	125	11.98%	1.85%	50
250	125	11.98%	2.89%	20
250	250	12.04%	2.47%	50
250	250	12.04%	5.29%	20

Tabella 5.4: Phinyomark con 9 classi

Finestra (ms)	Offset (ms)	Errore LDA	Errore NN	Hidden Neurons
500	62	35.06%	1.62%	50
500	62	35.06%	2.12%	20
500	125	34.73%	3.22%	50
500	125	34.73%	6.39%	20
500	250	34.43%	7.44%	50
500	250	34.43%	7.14%	20
250	62	38.2%	3.65%	50
250	62	38.2%	5.58%	20
250	125	38.47%	5.29%	50
250	125	38.47%	6.19%	20
250	250	38.42%	12.87%	50
250	250	38.42%	11.64%	20

Tabella 5.5: Phinyomark 2 con 9 classi

Finestra (ms)	Offset (ms)	Errore LDA	Errore NN	Hidden Neurons
500	62	8.41%	0%	50
500	62	8.41%	0.05%	20
500	125	8.44%	0.31%	50
500	125	8.44%	0%	20
500	250	9.43%	0%	50
500	250	9.43%	3.04%	20
250	62	10.77%	0.09%	50
250	62	10.77%	0.05%	20
250	125	10.97%	0.28%	50
250	125	10.97%	0.19%	20
250	250	11.14%	0.73%	50
250	250	11.14%	1.46%	20

Tabella 5.6: Hudgins con 5 classi

Finestra (ms)	Offset (ms)	Errore LDA	Errore NN	Hidden Neurons
500	62	1.91%	0.05%	50
500	62	1.91%	0.05%	20
500	125	2.13%	0%	50
500	125	2.13%	0.1%	20
500	250	1.74%	0%	50
500	250	1.74%	2.02%	20
250	62	3.94%	0.23%	50
250	62	3.94%	0.33%	20
250	125	3.52%	0.47%	50
250	125	3.52%	1.03%	20
250	250	4.19%	0.55%	50
250	250	4.19%	1.46%	20

Tabella 5.7: Phinyomark con 5 classi

Finestra (ms)	Offset (ms)	Errore LDA	Errore NN	Hidden Neurons
500	62	13.2%	0.42%	50
500	62	13.2%	0.84%	20
500	125	12.67%	1.35%	50
500	125	12.67%	0.83%	20
500	250	13.24%	4.66%	50
500	250	13.24%	2.83%	20
250	62	15.74%	2.67%	50
250	62	15.74%	1.59%	20
250	125	15.84%	2.43%	50
250	125	15.84%	3.27%	20
250	250	15.73%	5.1%	50
250	250	15.73%	3.83%	20

Tabella 5.8: Phinyomark 2 con 5 classi

Finestra (ms)	Offset (ms)	Errore LDA	Errore NN	Hidden Neurons
500	62	0.51%	0%	50
500	62	0.51%	0%	20
500	125	0.75%	0%	50
500	125	0.75%	0%	20
500	250	0.85%	0%	50
500	250	0.85%	0%	20
250	62	1.21%	0%	50
250	62	1.21%	0%	20
250	125	1%	0%	50
250	125	1%	0.12%	20
250	250	1.64%	0%	50
250	250	1.64%	0.24%	20

Tabella 5.9: Hudgins con 4 classi

Finestra (ms)	Offset (ms)	Errore LDA	Errore NN	Hidden Neurons
500	62	0%	0%	50
500	62	0%	0%	20
500	125	0%	0%	50
500	125	0%	0%	20
500	250	0%	0%	50
500	250	0%	0%	20
250	62	0.14%	0%	50
250	62	0.14%	0%	20
250	125	0.09%	0%	50
250	125	0.09%	0%	20
250	250	0.06%	0%	50
250	250	0.06%	0%	20

Tabella 5.10: Phinyomark con 4 classi

Finestra (ms)	Offset (ms)	Errore LDA	Errore NN	Hidden Neurons
500	62	1.93%	0.07%	50
500	62	1.93%	0.14%	20
500	125	2.24%	0.27%	50
500	125	2.24%	0.27%	20
500	250	1.73%	0.53%	50
500	250	1.73%	0%	20
250	62	3.95%	0.12%	50
250	62	3.95%	0.43%	20
250	125	4.13%	1.46%	50
250	125	4.13%	0.12%	20
250	250	4.19%	0.95%	50
250	250	4.19%	0.71%	20

Tabella 5.11: Phinyomark 2 con 4 classi

Dal dettaglio è subito evidente come il set di caratteristiche *Phinyo-*

mark2, composto dalle feature *MYOP (0,05), ZC, WAMP (0.01), MAV*, è da scartare, avendo le performance di classificazione peggiori.

Tra il feature set di *Hudgins* e di *Phinyomark*, composto dalle feature *SampEn , CC ordine 4, RMS , WL*, valutando l'errore nelle reti neurali, si hanno delle performance molto simili.

In caso di reti neurali la scelta dell'insieme delle caratteristiche più appropriato si è basata quindi unicamente sul miglior tempo di calcolo delle feature.
Di conseguenza l'uso di Hudgins, che è un set di caratteristiche meno complesso e largamente utilizzato in ambito di protesi mioelettriche, è risultata la soluzione migliore. Infatti fornisce performance di classificazione attorno al 98% minimo, che è un dato vicino all'ottimo, impiegando però solo $1ms$ per essere calcolato. Phinyomark d'altra parte è da scartare perché richiede $1100ms$ per calcolare un segmento (con una finestra di 500 ms): considerando che queste performance sono state profilate su un processore $i74770k$, il risultato sarà sicuramente peggiore sull'*Intel Edison*, e già così la tempistica è fuori dal limite fissato per consentire un utilizzo fluido della protesi.

Per questo motivo il classificatore scelto ha queste caratteristiche:

- Segmentazione di 500 ms (500 campioni) e offset di 62 ms (63 campioni)
- Classificatore a rete neurale con 50 neuroni
- Set delle caratteristiche di Hudgins

Si è evidenziato nei test effettuati che l'offset della finestra di acquisizione è ininfluente per quanto riguarda le performance di classificazione con LDA, mentre sul classificatore a rete neurale il peggioramento, seppur minimo, è visibile poiché la diminuzione della distanza tra le finestrature nel dataset fornisce alla rete neurale più campioni in fase di training: passare da un offset di 125 ad un offset di 62 ci fornisce il doppio delle caratteristiche da analizzare. Come è risaputo la rete neurale ha delle performance monotone decrescenti con la dimensione del dataset fornito. In base a queste osservazioni si ritiene che la dimensione del fattore di slittamento della finestratura possa essere scelta arbitrariamente rispettando solo dei vincoli di carico computazionale. Per quanto riguarda il tipo di classificatore scelto bisogna però essere consapevoli di perdere un'importante proprietà dell'LDA, ovvero la distanza massima del confine decisionale dai membri delle classi.

5.3 Studio sull'intero segnale e classificatori a due fasi

Come specificato in precedenza il segnale EMG è composto dalla fase di Rest, durante la quale, nel caso di articolazione alla spalla, il braccio si trova in posizione di riposo, dalla fase transiente (o burst), in cui l'arto inizia a muoversi, e dalla fase di steady, ossia quando rimane nella posizione target.

Finora il segnale EMG valutato è in realtà inteso come sola fase steady: il burst non è stato preso in considerazione.
Diversi studi utilizzano quest'ultima fase anziché quella di steady per cercare di anticipare la classificazione della classe di movimento e ridurre i tempi di risposta, ma il burst ha un andamento non costante, ed è per questo che le performance sono inferiori rispetto all'utilizzo della sola fase steady.

Un articolo di Hargrove Et Al. [54] dimostra che la combinazione di stato transiente e steady, nella fase di training, può migliorare l'usabilità del sistema di riconoscimento. Facendo ciò si riduce l'accuratezza del classificatore (steady: 97-100%, combinato: 92-96%), ma migliorano le performance real time. Partendo da questo presupposto un nuovo approccio viene descritto in questa tesi, basato sulla valutazione dell'intero segnale elettromiografico, al fine di migliorare l'utilizzo della protesi real time, obiettivo principale dello studio.

Lo studio di fattibilità è stato eseguito con lo strumento Matlab, per cui di seguito verranno descritti brevemente i due percorsi intrapresi.
I dettagli riguardanti il miglioramento delle performance verranno forniti nel capitolo *Implementazione su Dispositivo Indossabile* alla sezione dedicata.

In letteratura sono presenti esempi di classificatori composti, la composizione può avvenire in parallelo, aggregando poi i risultati in base alla loro confidenza; o possono essere composti in serie utilizzando l'uscita del primo classificatore come ingresso del classificatore nuovo. Questo approccio viene usato con successo componendo una *SOFM* in serie con una rete neurale: in questo esempio la *SOFM* produce un mapping non lineare tra i valori in ingresso e uno spazio bidimensionale con cardinalità pari al numero di nodi che la compongono, la rete neurale avrà come ingresso il pattern di attivazione della mappa delle feature e produrrà il suo output di classificazione.

L'approccio presentato di seguito è innovativo in quanto non si basa sull'analisi in cascata dello stesso segnale che fluisce attraverso diversi strumenti di machine learning, ma utilizza diversi classificatori con competenze specifiche, che collaborano alla determinazione della classe globale. Come prima cosa occorre dividere lo spazio delle classi in gruppi che hanno delle specificità comuni, verrà spiegato più avanti la divisione effettuata in questo studio

di tesi. Il fatto di isolare le classi in gruppi e responsabilizzare un classificatore rispetto ad un altro per la loro identificazione consente di ottenere un risultato molto più accurato in quanto si eliminano delle combinazioni di classi difficili da predire. Prendendo ad esempio il segnale elettromiografico, posso dividere il segnale in rest e steady, ed un classificatore può essere molto accurato nel separare la fase di steady rispetto alla fase di rest (basti pensare che solamente la caratteristica MAV è generalmente sufficiente ad effettuare questa distinzione), ed essere molto accurato nel discriminare il movimento effettuato una volta che è sicuro che il segmento appartenga alla fase di rest.

L'altra novità sostanziale è di dividere i gruppi di classi in modo gerarchico, un approccio classico effettuerebbe la classificazione con entrambi i classificatori per poi mettere insieme i risultati; ma se analizziamo bene il caso specifico del segnale EMG, evidenziamo una dipendenza tra i due classificatori: il classificatore rest-steady può influenzare il classificatore delle otto classi di movimento, mentre non ha senso sostenere il contrario. Da queste osservazioni è nata l'idea di un classificatore a fasi: ogni fase emette una variabile di controllo la quale viene utilizzata per scegliere quale classificatore attivare nella fase successiva, fino all'ultima fase.

Nel primo tipo di classificatore a due fasi, la prima fase discrimina tra burst-steady-rest mentre la seconda fase esegue il classificatore steady, il classificatore burst o nessuna classificazione a seconda della predizione della prima fase. Nel secondo tipo di classificatore, la prima fase discrimina tra segnale attivo-rest mentre la seconda esegue la classificazione del segnale attivo o nessuna classificazione a seconda della predizione della prima fase. L'output totale del classificatore può essere visto come la traccia di classificazione di ogni fase, o in pratica può essere ottenuto concatenando le variabili di controllo prodotte da ogni fase.
Si noti come l'output del classificatore non sia omogeneo ma assuma un formato diverso a seconda del percorso di classificazione che è stato attivato all'interno delle fasi.

5.3.1 Classificatore a due fasi con valutazione separata di steady e burst

In questo tipo di classificatore il segnale nella prima fase viene inizialmente separato nelle tre parti che lo compongono: Rest, burst e steady.
Il dataset viene quindi suddiviso in tre insiemi a seconda della fase di appartenenza, e su ognuno di essi vengono calcolate le caratteristiche di Hudgins e la PCA. Infine verranno applicati tre classificatori FFNN a 50 neuroni diversi: il primo andrà a classificare la fase corrispondente (classificatore STR)

e gli altri due verranno utilizzati rispettivamente per la fase di burst e per la fase di steady 5.4.

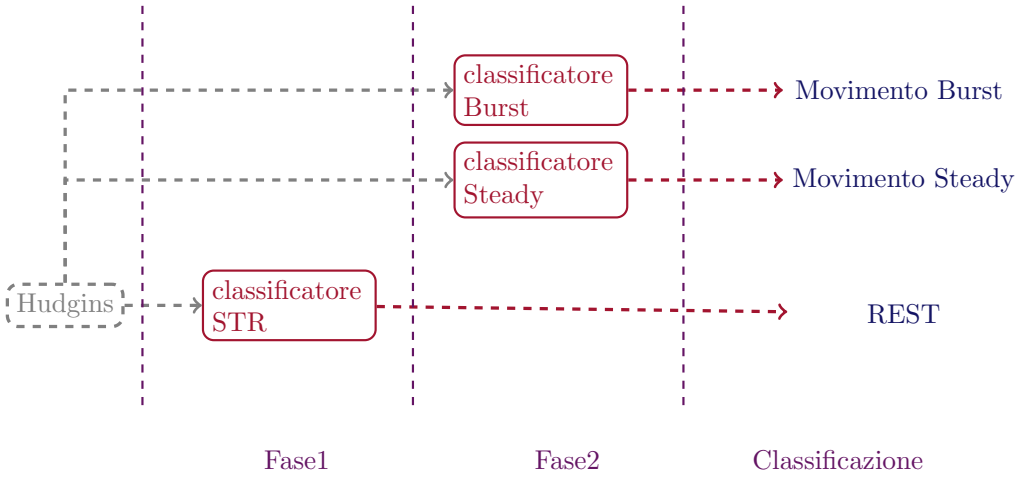


Figura 5.4: Flusso di lavoro del classificatore a due fasi con valutazione separata di steady e burst. La prima fase classifica il segnale in attivo di steady, attivo di burst o di riposo e nel caso sia attivo la seconda fase classifica il tipo di movimento osservato. All'interno della seconda fase viene invocato il classificatore corrispondente alla fase riconosciuta nella prima fase

I risultati teorici possono essere osservati in figura 5.5.

Per il classificatore che distingue le tre fasi Steady-Transiente-Rest l'accuratezza è del 95,9%, mentre per il classificatore della fase di burst è del 95,17% e quello della fase steady è del 99,9%.
Visti i risultati accettabili, in linea con le percentuali del lavoro di Hargrove Et Al. [54] precedentemente citato, questo primo tipo di classificatore è stato valutato accettabile per l'implementazione sul dispositivo indossabile.
Difatti una percentuale così bassa per il classificatore della fase transiente dipende dal fatto che questa fase del segnale contiene meno informazione rispetto alle altre.

5.3.2 Classificazione a due fasi con segnale combinato

In questo secondo tipo di classificatore le fasi di burst e steady vengono valutate come un unico segnale combinato, in modo tale da prelevare nella fase di training informazioni provenienti dall'intero segnale utile alla classificazione.

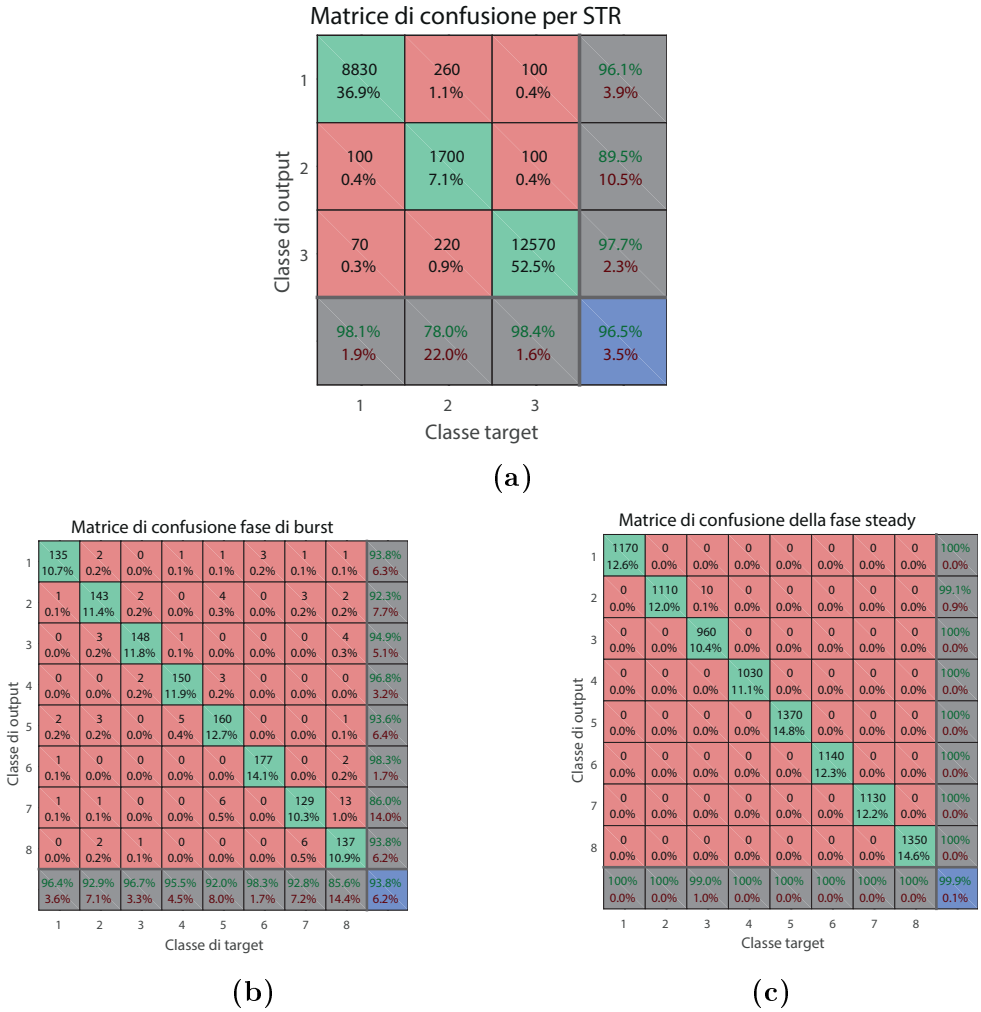


Figura 5.5: Matrici di confusione dei tre classificatori che compongono il classificatore a due fasi con valutazione separata di steady e burst. **(a)** Matrice di confusione per il riconoscimento delle 3 fasi Seady-Transient-Rest (accuratezza del 96.5%). **(b)** Matrice di confusione per la fase Burst (accuratezza del 92.8%).**(c)** Matrice di confusione per la fase di Steady (accuratezza del 99.9%).

Questo, in una situazione real time, permette di avere maggiori informazioni sull'intero segnale, riuscendo quindi a valutare prima il dato in arrivo.

In figura 5.6 è possibile osservare che, similmente al classificatore a due fasi descritto nella sezione precedente, inizialmente viene svolta la prima fase di discriminazione tra rest, steady e burst; successivamente viene utilizzato un classificatore in grado di classificare l'intero segnale.

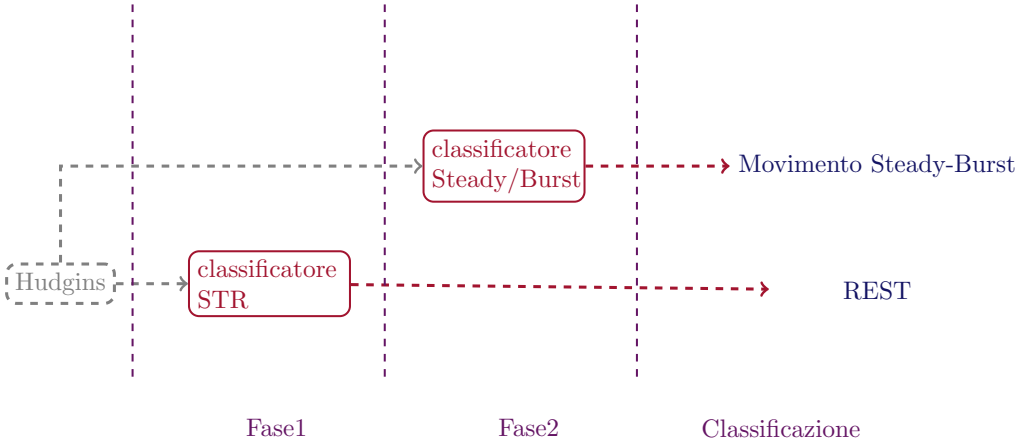


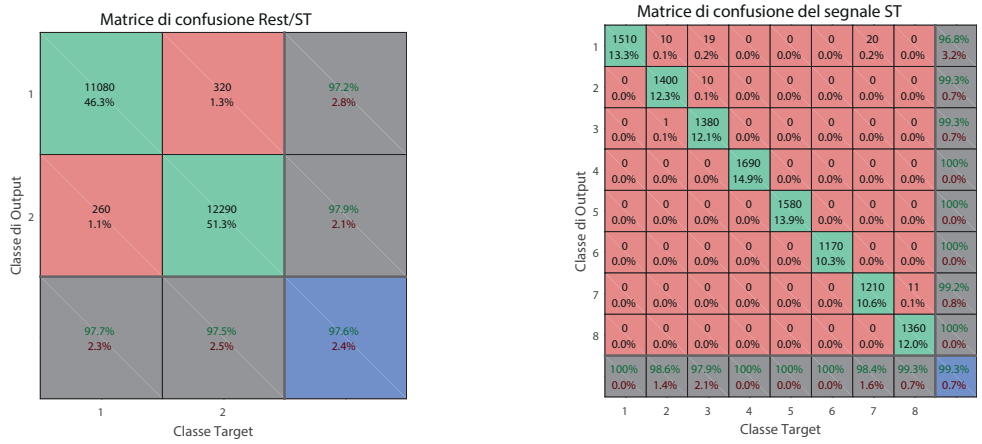
Figura 5.6: Flusso di lavoro del classificatore a due fasi con segnale combinato. L'azione della prima fase separa il segnale in segnale di riposo e segnale attivo e se il segnale viene classificato come attivo, la seconda fase classificherà il movimento osservato

Per questo tipo di classificatore a due fasi i risultati, come nel classificatore con valutazione separata di steady e burst, sono descritti in figura 5.7.

Le matrici di confusione indicano le seguenti percentuali di successo: per il classificatore STR il valore è pari a 97,6% mentre per quello che valuta il segnale combinato steady-burst è di 99,3%.
Questi risultati mostrano un aumento delle performance rispetto al classificatore a due fasi con valutazione separata di burst e steady. Inoltre sono confrontabili con lo studio sopracitato [54], per cui è risultato interessante valutare questo tipo di percorso in un dispositivo indossabile.

5.4 Studio sull'eliminazione dei canali per il classificatore a reti neurali

Nella sezione 4.2 presente al capitolo Analisi del Dataset sono stati valutati come canali peggiori a livello di performance il 2 e l'8, che rispettivamente sono posizionati sul Grande Pettorale Sternale e sul Gran Dorsale. Valutando le tabelle della suddetta sezione, sono stati effettuati dei test di performance, sia sui singoli canali che su un loro sottoinsieme, per capire fino a quando è



(a) **(b)**

Figura 5.7: Matrici di confusione dei due classificatori. **(a)** Valutazione Rest-Segnale combinato(steady e burst valutati insieme) (accuratezza del 97.6%). **(b)** Valutazione segnale combinato (accuratezza del 99.3%).

possibile diminuirne e avere comunque delle performance accettabili in fatto di accuratezza.

Per i test con l'utilizzo di un solo canale 5.8 è stato utilizzato il classificatore a rete neurale che utilizza la sola fase steady, perché più semplice. Infatti ciò che si vuole valutare non è l'accuratezza rispetto al classificatore, ma semplicemente la differenza di performance tra i canali.

Come si può notare dalla figura 5.8, i canali 2 e 8, presi da soli, hanno le performance peggiori, e ciò conferma quanto detto durante la valutazione dei canali. Sulla base di questi valori, e con le informazioni ottenute durante lo studio appena citato, sono stati testati diversi sottogruppi di canali, levando via via quelli che statisticamente avevano performance peggiori, sia dal punto della classificazione ottenuta con il singolo canale, sia valutando la capacità di discriminazione del canale tra i movimenti, sia la variabilità dei valori assunti, sia la tendenza ad avere per uno stesso movimento il più piccolo range di valori possibili.
Secondo le valutazioni precedenti, i canali 1, 3, 6 e 7 sono ritenuti i migliori per quanto riguarda la discriminazione dei movimenti e/o per avere dei valori limitati per il singolo movimento, dando la possibilità di distinguerlo meglio rispetto agli altri.
A valle di queste argomentazioni sono stati confrontati anche i dati relativi agli errori di classificazione del canale preso singolarmente, e così facendo è

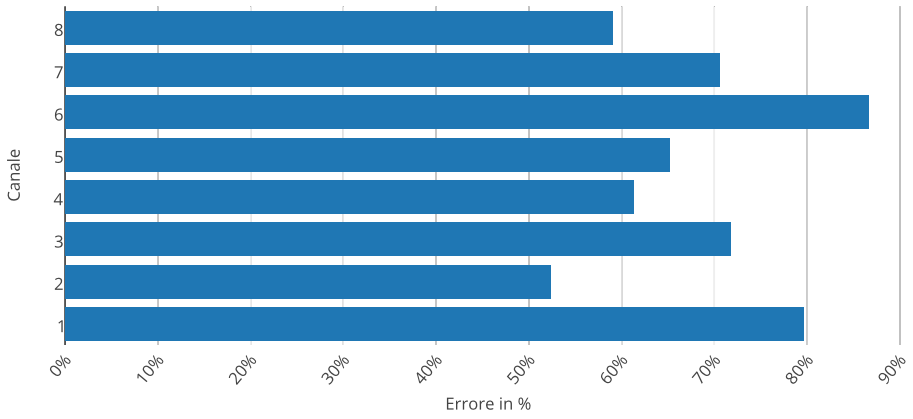


Figura 5.8: Percentuale di successo di classificazione per ciascun canale preso singolarmente.

stato scelto il canale da eliminare passo dopo passo. Per i test che riguardano i sottoinsiemi di canali sono stati utilizzati sia il classificatore FFNN con la sola fase steady, sia quello che utilizza sia il burst che lo steady, per poter fare un paragone con una configurazione utilizzabile nella realtà. Nella tabella 5.12 sono riportati i risultati.

Innanzitutto bisogna specificare che nel caso delle reti neurali le performance sono fortemente influenzate dall'inizializzazione dei neuroni della rete. Tale fase viene effettuata in maniera casuale, per cui i valori percentuali presenti possono variare di un fattore $\pm 0,5\%$. Premesso ciò, è facilmente osservabile che il classificatore FFNN che utilizza solo la fase steady ha performance maggiori rispetto a quello con l'intero segnale, e raramente scende al di sotto del 98%. Osservando nell'insieme i risultati però sicuramente è possibile ridurre almeno tre canali (il 2, il 4 e l'8) per avere performance non inferiori al 99%

5.5 Fase di training implementata

La fase di training viene effettuata in tempi precedenti all'uso vero e proprio della protesi.
Solitamente la persona con arto amputato, una volta valutata la protesi più

Canali	Performance con FFNN (burst+steady)	Performance con FFNN (solo steady)
1, 2, 3, 4, 5, 6, 7	98,95%	100%
1, 3, 4, 5, 6, 7, 8	99,385%	99,676%
1, 3, 4, 5, 6, 7	99,3%	99,89%
1, 3, 5, 6, 7	99,6%	100%
1, 3, 4, 6, 7	99,3%	99,8%
1, 3, 4, 5, 7	98,68%	99,6%
1, 3, 6, 7	98,2%	99,2%
1, 3, 5, 7	98,33%	99,67%
1, 3, 7	96,7%	97,6%
1, 3, 6	97,2%	99,4%

Tabella 5.12: Percentuali di accuratezza per i due classificatori FFNN di diversi sottoinsiemi di canali (i valori possono variare di $\pm 0,5\%$ per l'inizializzazione dei neuroni).

adatta, deve recarsi in strutture predisposte per tale scopo.
Negli ultimi anni si sta cercando di rendere la fase di training autonoma da parte del futuro portatore di protesi, sia per abbassare i costi dovuti alla formazione e all'assunzione di personale per svolgere questa mansione, sia per dare la possibilità all'utente di scegliere i tempi di apprendimento, in un ambiente familiare e più comodo.

L'interfaccia utente

L'idea iniziale in questo lavoro di tesi era quella di implementare un'interfaccia user-friendly per la fase di traning autonoma, ma non avendo inizialmente una strumentazione consona per la registrazione di segnale sEMG e avendo a disposizione un dataset limitato, l'interfaccia implementata, molto semplice, è rivolta a un ipotetico tecnico 5.9 che ha la possibilità di scegliere:

- il numero di soggetti (da 1 a 8) tra il dataset a disposizione;

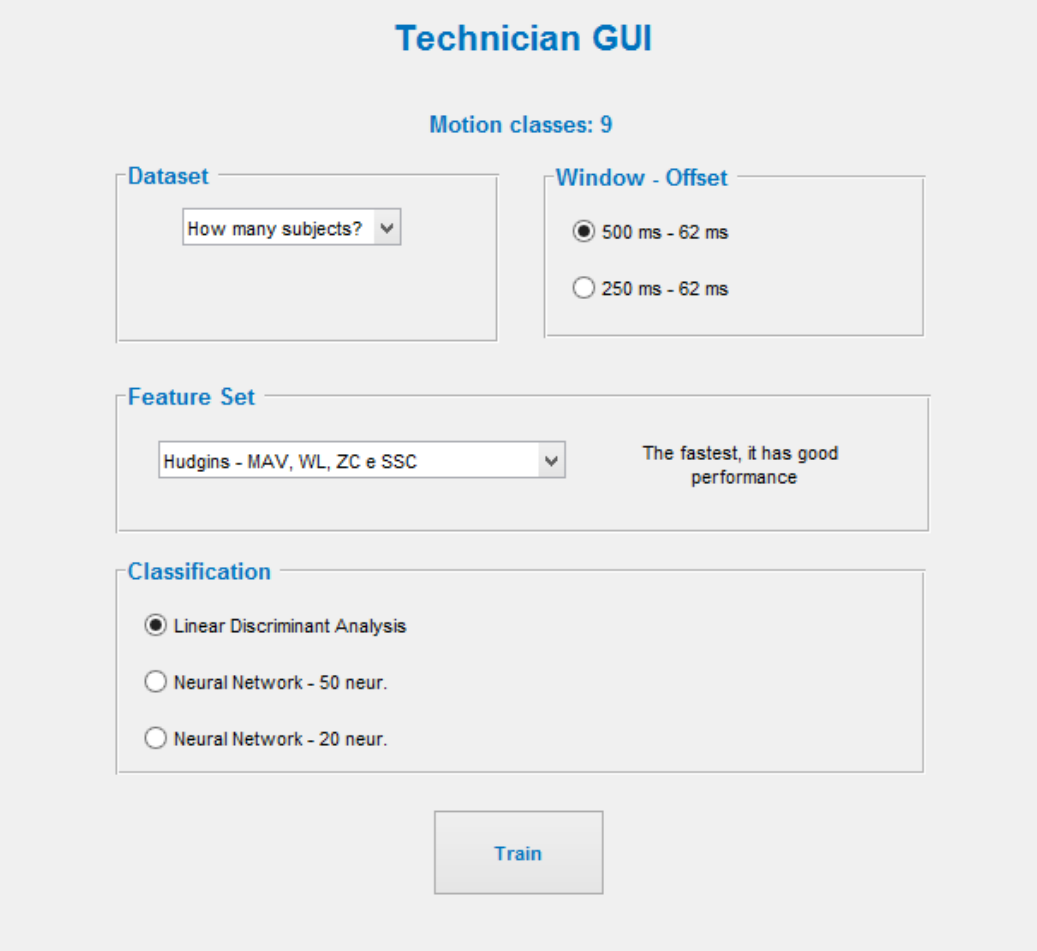


Figura 5.9: GUI iniziale.

- la segmentazione per l'analisi del segnale tra le due possibilità con performance più alte, per cui lunghezza di finestra di 500 o 250 ms e offset di 62 ms;
- il feature set da utilizzare, tra Hudgins, Phinyomark e Phinyomark2 (Tabella 2.1);
- il tipo di classificatore, tra LDA, NN con 50 neuroni e NN con 20 neuroni.

Inoltre viene visualizzato il numero di classi di movimento che il classificatore è in grado di discriminare. In particolare vengono sempre usate le nove classi di movimento descritte da Rivela e Scannella [1]. Questa scelta è

stata presa perchè in una protesi bisogna dare la possibilità all'utente di muoversi il più liberamente possibile, e ciò si fa cercando di aumentare il numero di movimenti riconoscibili dal classificatore (e quindi poi attuabili da un controllore).

Una volta eseguito il codice verrà visualizzata una finestra in cui ci saranno le percentuali di errori di training e di testing e le relative matrici di confusione 5.10.

Figura 5.10: GUI dei risultati.

5.5.1 Il codice

Una volta che l'utente sceglie i parametri, il codice Matlab effettuerà la vera e propria fase di training.
Inizialmente, partendo dal segnale EMG e dall'informazione della cinematica del movimento del dataset di Rivela e Scannella, verranno individuati la fase di inizio e di fine steady.
Individuate queste fasi stabili, viene applicata la segmentazione del segnale, rispettando la lunghezza e la sovrapposizione di finestra scelti dall'utente.
Infine vengono calcolate sul segnale segmentato le feature appartenenti al set prescelto.
Questo iter viene eseguito per ogni movimento e per ogni soggetto.

Successivamente i dati devono essere preparati per essere poi passati al classificatore.

Ovviamente la quantità di caratteristiche calcolate ha una dimensionalità elevata, per cui viene applicata la PCA.
Questa fase di *riduzione delle caratteristiche* serve per diminuire la complessità di elaborazione che il classificatore dovrà affrontare in seguito. Il numero di componenti principali n su cui verranno proiettati tutti i dati ottenuti finora dipende dall'insieme di caratteristiche in esame.
Sperimentalmente il set Hudgins ha un numero di componenti pari a 17, Phinyomark a 56 e Phinyomark2 a 24.
Infine verrà effettuata la classificazione mediante il classificatore precedentemente selezionato e le informazioni finali verranno poi trasferite al sistema indossabile Edison che, a partire dai dati di training, potrà identificare le classi di movimento a run-time, allenando la propria rete neurale a riconoscere tali classi.

Il codice sorgente è disponibile per il download come indicato nell'appendiceA.

5.5.2 Lo scambio dei dati

Una volta effettuata la fase di training è necessario determinare un metodo per trasferire l'informazione prodotta da Matlab a Edison. Una volta ottenuta la classificazione, alcuni parametri verranno memorizzati in un file *JSON* per essere poi trasferiti al secondo sistema. Il motivo per cui è stato scelto questo formato per lo scambio di dati e che oggi è ampiamente usato in svariate applicazioni. Inoltre è maggiormente leggibile da un utente umano rispetto ad altri, quali ad esempio XML, ed è particolarmente semplice da utilizzare.

Il modello deve contenere le informazioni relative alle matrici della rete neurale o del classificatore LDA, le informazioni relative alle classi di movimento e le informazioni relative a alla forma della rete/matrice. Di seguito verrà descritto il modello adottato e nel capitolo *Implementazione su Dispositivo Indossabile* verranno forniti i dettagli del contenuto di ogni elemento.

Listing 5.1: Modello Json per il trasferimento dei dati

```
1 {
2   "weight" :[... ],
3   "bias" :[... ],
4   "xoff" :[... ],
5   "pca" :[... ],
6   "xgain" :[... ],
7   "ym" :-1.000000,
```

```
8   "n_sample" :32,
9   "input" :17,
10  "output" :8,
11  "hidden" :50,
12  "mu" :[... ],
13  "sigma" :[... ],
14  "type" :"NeuralNetwork",
15  "ClassLabel" :["Fl 45","Fl 90","Fl 110","I -30","Ab 45","Ab 9
        0","El 45diagonale","El 90diagonale"]
16 }
```

6. Implementazione su Dispositivo Indossabile

Beauty is more important in computing
than anywhere else in technology.
Beauty is the ultimate defence against
complexity.

David Gelernter

In questo capitolo viene illustrata la progettazione e l'implementazione del dispositivo indossabile. Molte delle problematiche relative alle prestazioni di classificazione sono state affrontate nei capitoli precedenti, in particolare nel capitolo *Pattern Recognition per il segnale EMG* alla sezione 2.3 vengono analizzati due classificatori basati sulle reti neurali e un classificatore basato su un sistema lineare. Nell'implementazione del dispositivo indossabile è stato usato il classificatore *FNN50* (Rete neurale di tipo feed forward con 50 neuroni), il quale si è imposto sugli altri per quanto riguarda la correttezza nel classificare il segnale elettromiografico. Il classificatore *FNN50* è particolarmente indicato per la classificazione in tempo reale e per la sua velocità nel produrre il segnale in uscita. *Hudgins* è stato mantenuto come metodo di estrazione delle caratteristiche, al quale è stata applicata la riduzione delle caratteristiche basata sulla analisi delle componenti principali e la normalizzazione seguendo come riferimento i test effettuati con il software implementato in Matlab.

Nel software di addestramento sono stati analizzati tre sistemi di classificazione compositi, presentando le loro performance teoriche; il percorso a classificatore singolo e i due percorsi a doppia fase sono stati implementati sul dispositivo indossabile e testati con dei tracciati EMG di prova per confrontare i risultati di classificazione con i risultati teorici esposti nei capitoli precedenti.

Il software è stato pensato per essere sia facilmente espandibile sia performante, usando il paradigma funzionale unito al paradigma della programmazione orientata agli oggetti. Questi due paradigmi sono stati scelti perché entrambi aiutano la manutenzione del codice e sono ricchi di pratiche consolidate per ottenere un codice di qualità. Il paradigma di programmazione orientata agli oggetti viene utilizzato per la struttura di gestione, mentre il paradigma funzionale è stato usato per gli algoritmi di basso livello come i classificatori e il calcolo delle caratteristiche. Questa scelta consente di rifinire la parte funzionale implementando le singole funzioni come moduli compilati, scritti in *C* e poi interfacciati con *Python*, il che consente di snellire i colli di bottiglia dovuti all'accesso a grandi porzioni di dati.

6.1 Librerie Utilizzate e Dipendenze

La scelta di *Python* per l'implementazione della maggior parte dei moduli del sistema è stata influenzata dal rilievo che questo linguaggio di programmazione ha all'interno della comunità accademica e della progettazione di software ingegneristico. Questo successo è dovuto in gran parte alla presenza di performanti librerie open-source per il calcolo numerico. Non è stata presa in considerazione solo la presenza di librerie esistenti ma anche le possibilità di estensione del linguaggio; *Python* integra al suo interno numerosi meccanismi di espansione e interoperabilità con altri linguaggi. Alcuni dei più usati sono: *Iron Python* che consente di interfacciarsi con il *.net Framework* di Microsoft, *Cython* che consente di compilare il codice in librerie *C* e *C API* ed espone direttamente i tipi di *Python* in *C*.

LibMRAA è una libreria *C/C++* con wrapper in *Python* e *Javascript* fornita da Intel per consentire ai dispositivi embedded come *Edison* e *Galileo* di interfacciarsi con l'hardware. Nel nostro progetto è stata inclusa come libreria per l'utilizzo delle interfacce di comunicazione (*I2C*, *UART* e *SPI*), ma consente anche di accedere a tutte le interfacce *GPIO* presenti sulla board.

pySerial è un modulo che fornisce la connettività *UART*.

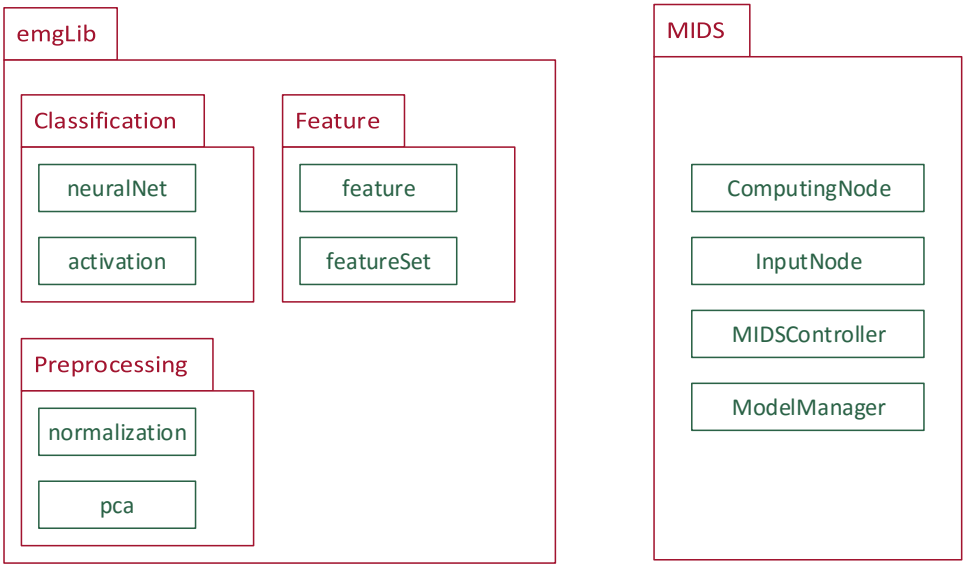


Figura 6.1: In questo diagramma è visibile la struttura software di Edison: composta dal package emgLib che contiene tutti gli strumenti per classificare il segnale *sEMG*, e dal package MIDS che contiene le strutture di controllo del dispositivo.

numpy è un insieme di moduli per il calcolo scientifico; la principale funzionalità è fornire gli *ndarray*, una versione di array ottimizzata e performante. Presenta anche librerie per l'integrazione con *C/C++*.

6.2 Architettura Dispositivo Indossabile

La struttura software è divisa in moduli, ognuno con le sue competenze specifiche, e la struttura completa può essere vista nello schema in figura 6.1. Il progetto è diviso in due moduli: *MIDS* (*Movement Intent Detection System*) e *emgLib*.

La progettazione del modulo *emgLib* è una diretta conseguenza dei risultati ottenuti con il software *Matlab* e ne implementa fedelmente gli algoritmi e le funzioni senza variazioni, sia per mantenere il sistema omogeneo ma anche perché il software sul dispositivo indossabile è fortemente dipendente dal suo duale in *Matlab*. Si è cercato di svincolare il dispositivo indossabile dal software di apprendimento fornendo un formato di scambio del modello, il quale consente di cambiare i due componenti in modo invisibile all'altro,

creando una relazione di dipendenza dal formato di scambio e non tra i singoli componenti.

Per il modulo *MIDS* c'è stata più libertà di progettazione, non dovendo sottostare a nessun vincolo se non alla ricerca delle soluzioni migliori per raggiungere gli obiettivi fissati.

Di seguito verranno analizzati i moduli singolarmente, presentando gli oggetti e le funzioni contenute al loro interno.

6.3 MIDS Movement Intent Detection System

MIDS, è il modulo che ospita tutti gli automatismi necessari alla gestione ad alto livello del dispositivo. Per questo modulo si è scelto uno stile orientato agli oggetti dove il flusso di elaborazione è stato diviso in input e classificazione e per ognuno dei due è stata creata una classe specializzata. Le classi incaricate dell'acquisizione e della classificazione hanno un'interfaccia comune che viene gestita da una classe controllore come delineato in figura 6.2. Il modulo contiene la classe *ModelManager* la cui responsabilità è di importare il modello generato da *Matlab* e creare i classificatori con i parametri individuati dal software di apprendimento, utilizzando gli strumenti presenti nel modulo *emgLib*.

Le classi presenti nel modulo sono:

MIDSController : si occupa di gestire la comunicazione tra i nodi Input e Computing.

Input Node : si occupa di leggere i dati dalla seriale che comunica con la scheda *Eracle* e di preparare i segmenti da passare all'oggetto Computing.

Computing Node : gestisce il calcolo delle caratteristiche e della classificazione dei segmenti, comunicando il risultato di classificazione al MIDSController.

6.3.1 MIDSController

La classe *MIDSController* consente di eseguire la lettura dei campioni e la classificazione in modo indipendente, implementando il paradigma produttore e consumatore multithreading, in modo tale che la lettura successiva possa essere simultanea all'analisi del segmento attuale.
Questo meccanismo è stato ottenuto utilizzando due semafori specifici per due

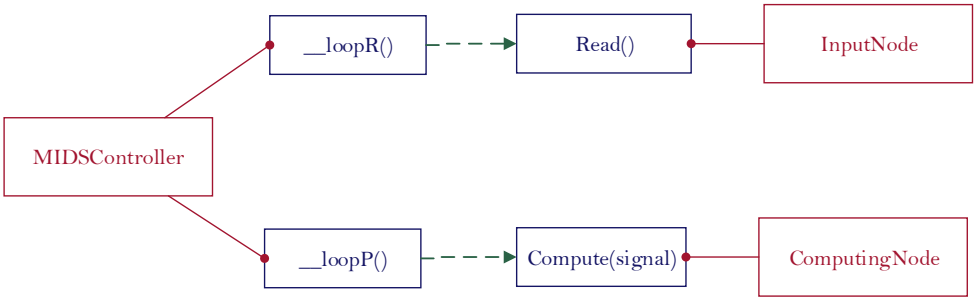


Figura 6.2: In questa figura viene mostrato come MIDSController interagisce con InputNode e ComputingNode. L'acquisizione e l'elaborazione sono implementate dagli oggetti InputNode e ComputingNode e gestite dall'oggetto MIDSController

risorse diverse. Come si può vedere nel listato seguente una risorsa chiamata `self.__handle` serve a sincronizzare il consumatore, forzandolo ad elaborare un segmento, solamente se questo è stato prodotto, mentre la seconda risorsa chiamata `self.__queueLOCK` serve a monitorare l'accesso al buffer, che può avvenire per mezzo di un solo thread alla volta, come specificato dal paradigma produttore e consumatore.

Listing 6.1: Sincronizzazione Thread

```
def __loopR(self):
        while(True):
            if(self.__stop.isSet()==True):
                self.__queueLOCK.acquire()
                self.__buffer.append(np.zeros(17))
                self.__queueLOCK.release()
                self.__handle.release()
                break
            #FUNCTION THAT READS THE DATA
            data = self.__reading()
            self.__queueLOCK.acquire()
            self.__buffer.append(data)
            self.__queueLOCK.release()
            self.__handle.release()
        return None

    def __loopP(self):
        while(True):
            if(self.__stop.isSet()==True):
                break
            #Always acquire
```

```
22            self.__handle.acquire()
23            self.__queueLOCK.acquire()
24            #FUNCTION THAT ANALIZE DATA
25            data = self.__buffer.popleft()
26            self.__computing(data)
27            self.__queueLOCK.release()
28        return None
```

Le funzioni sono implementate esternamente e passate in fase di inizializzazione al *MIDSController*, i callback vengono composti dal programma principale scegliendo la funzione più adatta del modulo *emgLib*. Come è possibile vedere nel listato di codice 6.2 il *MIDSController* può eseguire qualsiasi coppia di funzioni passata in fase di inizializzazione, così da poter svincolare la classe *MIDSController* dalle classi *InputNode* e *ComputingNode*.

Listing 6.2: Inizializzazione delle funzioni di lettura

```
1  def reading():
2      return iNode.input()
3  #end reading function
4
5  def computing(signal):
6      if(signal.shape!=(500,8)):
7      return
8      data = node.compute(signal)
9      me = max(data)
10     maxi =8-[i for i,x in enumerate(data) if x == me][0]
11     #Here is where comunication outside has to be implemented
12     if(maxi<8):
13         l = mm.getLabel(maxi,'second')
14         print "----Computed {}".format(l)
15     else:
16         x = 0
17         print "----Computed REST"
18 #end computing function
19
20 m = MIDS.MIDSController(reading,computing)
```

6.3.2 Model Manager

Il caricamento del modello viene gestito dall'oggetto *ModelManager* che si occupa di leggere un apposito file *Json*. Questo file è organizzato gerarchicamente: il primo livello contiene una lista di classificatori, ognuno dei quali darà vita ad un tipo di classificatore diverso; sarà il nodo responsabile del loro utilizzo, in questo caso il ComputingNode, a preoccuparsi di comporre

il percorso di classificazione e di conseguenza a scegliere tra una struttura semplice o un classificatore a due fasi.

All'interno di ogni classificatore troviamo i seguenti campi:

weight Contiene le matrici dei pesi della rete neurale. Queste sono linearizzate e concatenate in un'unica lista che può essere scompattata, conoscendo la dimensione dell'input, per ricostruire le tre matrici singole, del livello nascosto e dell'output, estraendole in quest'ordine dalla sequenza.

bias Contiene i vettori soglia da sottrarre ad ogni livello della rete neurale. In alcune implementazioni questi pesi sono inclusi nella matrice dei pesi della rete neurale, ma in questo lavoro si è voluta mantenere la coerenza con l'implementazione Matlab che invece tende a tenerli separati.

xoff Contiene il vettore dell'offset da sottrarre al segnale in ingresso al classificatore neurale. Insieme a *xgain* e *ym* fa parte dei parametri forniti alla funzione di adattamento dell'ingresso *mapMinMax*.

xgain Contiene il fattore di scala per il vettore degli ingressi della rete neurale.

pca Contiene la matrice $O \times R$ che riduce la dimensionalità dello spazio delle caratteristiche da O a R.

ym Contiene il segnale costante da aggiungere al vettore di ingresso dopo la moltiplicazione per il fattore di scala.

n_sample Corrisponde alla numerosità dello spazio delle caratteristiche prima della riduzione con l'analisi delle componenti principali.

input Corrisponde alla numerosità del vettore delle caratteristiche ridotto in ingresso al primo livello del classificatore.

output Corrisponde alla numerosità del vettore di classificazione dopo aver applicato la funzione *SoftMax*.

hidden Corrisponde al numero dei neuroni nel layer nascosto.

mu Corrisponde alla media estratta dal dataset con cui viene normalizzato ogni campione di dati secondo la funzione $n = (i - \mu/\sigma)$.

sigma Corrisponde alla deviazione standard σ del dataset con cui si effettua la normalizzazione.

type Indica la famiglia di classificatore utilizzata, in questa implementazione viene utilizzato solo il classificatore *FFNN50*. La proprietà avrà sempre il valore `'NeuralNetwork'`, ma è stato inserito per mantenere la compatibilità con eventuali versioni future.

ClassLabel Contiene la corrispondenza simbolica tra il vettore di output e le classi di movimento.

Il contenuto del file è stato pensato per essere espandibile, in modo tale da poter aggiungere nuove modalità di funzionamento senza compromettere quelle esistenti. Per esempio per mezzo del campo `'type'` è possibile inserire nuovi tipi di modelli generati da qualsiasi software in grado di generare un file *Json*.

Il file *Json* contenente il modello di classificazione può essere trovato nella sotto cartella model. Avviando il software è possibile indicare quale modello caricare.

6.3.3 InputNode

Questo nodo si occupa di acquisire il segnale proveniente dalla scheda di acquisizione e di comporre il segmento da analizzare concatenando le singole acquisizioni. Allo stato attuale il dispositivo comunica con la scheda di acquisizione attraverso una porta seriale utilizzando la libreria *pySerial*, ma sfruttando la libreria *mraa* é possibile aggiungere la compatibilità con sistemi *SPI*, *I2C*.

Esistono due implementazioni della classe *InputNode*, la prima riceve i dati da una scheda attraverso il protocollo seriale, l'altra implementazione legge i dati da un file *CSV (Comma Separated Value)* salvato sul dispositivo e lo analizza come se fosse stato acquisito da una scheda fisica. I test riguardanti la precisione di classificazione sono stati eseguiti utilizzando la seconda modalità per rendere le operazioni più veloci e più controllabili, mentre i test sulle performance temporali sono stati eseguiti con la scheda fisica per analizzare meglio i tempi di esecuzione in un ambiente il più verosimile possibile. Ogni implementazione del nodo di input deve implementare il metodo **`ndarray read()`** che fornisce un segmento ad ogni sua invocazione. Contenendo delle chiamate bloccanti che non restituiscono il controllo al codice chiamante fino alla produzione del dato, la funzione `read()` deve essere eseguita in un Thread separato, cosi da poter meglio sfruttare le caratteristiche del *SoC* multicore e non bloccare l'analisi del segmento acquisito in precedenza.

Il formato di acquisizione implementato è fortemente dipendente dal protocollo della scheda *Eracle*, che consiste in una sequenza di messaggi in formato stringa separati dal carattere di *carriage return*. Una singola acquisizione

è identificata dal messaggio `D:CH1 CH2 CH3`, dove `CH1 CH2 CH3` sono dei valori a doppia precisione corrispondenti ai tre canali con le unità separate dai valori decimali per mezzo del carattere '.'. Ogni 100 campioni inviati la scheda invia il messaggio `I:A B C` utile per effettuare delle sincronizzazioni o monitorare il funzionamento della scheda, che però non è stato necessario utilizzare nell'implementazione corrente.

Per dialogare con la scheda *Eracle* la seriale è stata settata con i seguenti parametri: *Baud* = 57600, *StopBit* = One, *Parità* = none, *Dimensione del Byte* = 8*bit*.

6.3.4 ComputingNode

La classe *ComputingNode* si occupa di analizzare il segmento acquisito dalla classe *InputNode* e applicare in successione l'estrazione delle caratteristiche, la riduzione dimensionale ed effettuare l'analisi attraverso il classificatore a rete neurale. Anche questa classe come la classe *InputNode* implementa una singola funzione che verrà richiamata dalla classe *MIDSController* in un Thread separato dalla funzione di lettura. Come spiegato in precedenza con questa scelta si vuole aumentare il parallelismo dell'algoritmo acquisizione-classificazione.

Listing 6.3: Codice della Classe ComputingNode

```
def __init__(self,net,pca,Mnet,Mpca,feature,mode='multi', *args,
    **kwargs):
        self.__net = net
        self.__pca_transform = pca
        self.__feature = feature

        self.__netSecond = Mnet
        self.__pca_transformSecond = Mpca
        self.__featureSecond = feature

        def ComputeTwo(signal):
            signal = self.__feature(signal)
            norm = self.__pca_transform(signal)
            stageOne = self.__net.update(norm)
            #calcolo secondo modello
            if(stageOne[0]<=stageOne[1]):
                #do other Classification
                computed_outcome = np.zeros(8)
            else:
                norm = self.__pca_transformSecond(signal)
                computed_outcome = self.__netSecond.update(norm)
            return computed_outcome
        #end Compute Single
```

```
23
24          def ComputeOne(signal):
25              signal = self.__feature(signal)
26              signal = self.__pca_transform(signal)
27              stageOne = self.__net.update(signal)
28              return computed_outcome
29          #end #end Compute Multi
30
31          if(mode=='multi'):
32              self.__do = ComputeTwo
33          else:
34              self.__do = ComputeOne
35          return super(ComputingNode, self).__init__(*args, **kwargs)
36
37      def compute(self,signal):
38          computed_outcome = self.__do(signal)
39          return computed_outcome
```

Come è visibile nel listato 6.3, la classe consente di utilizzare entrambi i classificatori a doppia fase presentati precedentemente, mentre la scelta della modalità di funzionamento viene impostata attraverso un parametro passato alla funzione di inizializzazione della classe.

6.4 emgLib

Il modulo *emgLib* è stato implementato seguendo il paradigma della programmazione funzionale e raccoglie tutti gli strumenti necessari alla classificazione del segnale *EMG* raggruppati in tre sottomuduli: *Feature*, *Preprocessing* e *Classification*.

6.4.1 Feature

Il modulo *Feature* contiene le implementazioni dell'estrazione delle caratteristiche. L'insieme di caratteristiche scelto per il dispositivo finale è stato *Hudgins* che è composto da *MAV*, *WL*, *ZC*, *SSC* per un totale di 32 caratteristiche su un segmento da otto canali, ridotte poi attraverso l'analisi delle componenti principali a 17 per poter essere analizzate dal classificatore *FFNN500*.

La prima implementazione utilizzava solo *Python* avvantaggiandosi delle performance del pacchetto *numpy*, il quale non è stato sufficiente ad ottenere un codice abbastanza veloce per gli standard di questo lavoro.

Listing 6.4: Prima implementazione Hudgins

```
def MAV(vect,faxis=0):
    return np.mean(np.absolute(vect),axis=faxis)
def WL(vect,faxis=0):
    return np.sum(np.absolute(vect[1:]-vect[:-1]),axis=faxis)
def ZC(vect):
    r = np.multiply(vect[1:,:],vect[:-1,:])
    temp = map(lambda x: 1 if x<0.0 else 0,r.flat)
    temp = np.array(temp).reshape((499,8))
    r3 = np.sum(temp,axis=0)
    return r3
def SSC(vect,th):
    r = np.multiply(vect[1:-1,:]-vect[:-2,:],vect[1:-1,:]-vect[2:,:])
    temp = map(lambda x: 1 if x>=th else 0,r.flat)
    temp = np.array(temp).reshape((498,8))
    r3 = np.sum(temp,axis=0)
    return r3
```

Questa prima implementazione è risultata essere il collo di bottiglia dell'intero sistema: profilando *Hudgins* in *Python* sul dispositivo Edison si è registrato un tempo di esecuzione di $104ms$, tempo per cui si rendeva critico il rispetto del vincolo real time, in quanto l'implementazione del solo classificatore si avvicinava ai $300ms$ limite. Molte delle funzioni della libreria *numpy*, come quelle utilizzate nel codice6.4, sono implementate in linguaggio *C* e successivamente incapsulate in un modulo Python, il che le rende adatte ad un utilizzo in tempo reale. Però alcune funzioni utilizzate nel calcolo delle caratteristiche *SSC* e *ZC* (da sole richiedevano $101ms$) vengono continuamente analizzate dal parser Python e non possono sfruttare la piena velocità fornita dalle implementazioni in *C*.

Per questo motivo è stata prodotta una seconda versione ottimizzata dell'insieme delle caratteristiche di Hudgins la quale riesce ad estrarre le 32 caratteristiche in meno di mezzo millisecondo.

La funzione di estrazione ottimizzata è stata implementata in linguaggio *C* in modo da analizzare tutto il segmento con codice compilato e quindi immune all'overhead che il codice Python produce. Per rifinire questa parte del classificatore, sono state valutate due opzioni: la prima consisteva nell'utilizzare *Cython*, la seconda nell'utilizzo delle *C API*.

Cython è uno strumento che può compilare il linguaggio Python in codice *C*. Questi moduli vengono scritti in un linguaggio Python, con alcune estensioni che lo modificano pesantemente trasformandolo quasi in un nuovo linguaggio, il quale viene poi trasformato da codice ad alto livello a codice macchina.

C API è una raccolta di librerie, utilizzabili in *C/C++*, che consentono di creare e utilizzare oggetti Python all'interno del codice *C*, il quale può essere

poi compilato e incluso in un modulo *Python* per mezzo della funzionalità contenuta nel modulo *Extensions*.

Le due tecniche sono entrambe valide, ma per questa tesi si è scelto di utilizzare la seconda in quanto molto più portabile della prima: i moduli scritti con *C API* possono essere compilati per qualsiasi piattaforma in cui esista un ambiente Python e un compilatore *C*, senza dover dipendere da altre strutture come ad esempio il linguaggio di estensione *Cython*. Inoltre il codice *Cython* risulta essere molto meno performante di quello scritto con le *C API*, dato che il linguaggio *C* mette nelle mani dello sviluppatore la completa gestione del codice; al contrario *Cython* non ha una corrispondenza diretta con il codice macchina e quindi risulta essere meno performante se non vengono prese le necessarie precauzioni.

Il codice ottimizzato della funzione Hudgins utilizza le librerie *C API* fornite da numpy, che espongono la classe *ndarray*. Come si può vedere nel listato seguente, l'*ndarray* viene utilizzato come interfaccia con il resto del codice Python:

Listing 6.5: Implementazione Hudgins C API

```
1
2  #include <Python.h>
3  #include <math.h>
4  #include <numpy/arrayobject.h>
5
6  static PyObject* hudgins(PyObject* self, PyObject* args)
7  {
8
9    PyArrayObject *in_array;
10   PyArrayObject *out_array;
11
12   /*  parse single numpy array argument */
13   if (!PyArg_ParseTuple(args, "O!", &PyArray_Type, &in_array))
14     return NULL;
15
16   /*  construct the output array, like the input array */
17   npy_intp shape = 32;
18   out_array = (PyArrayObject *)PyArray_SimpleNew(1,&shape, NPY_DOUBLE);
19   if (out_array == NULL)
20     return NULL;
21
22   PyArray_FILLWBYTE(out_array, 0);
23
24   int samples = *(in_array->dimensions);
25   int channels = *(in_array->dimensions + 1);
26
27   double mav = 0.0;
```

```
28  double wl = 0.0;
29  double zc = 0.0;
30  double ssc = 0.0;
31
32  int channel_stride = 8;
33  for (int c = 0; c < channels; ++c) {
34    mav = 0.0;
35    wl = 0.0;
36    zc = 0.0;
37
38    double d = *(double*)(in_array->data + 0 * in_array->strides[0] + c *
          in_array->strides[1]);
39    mav += fabs(d);
40
41    for (int j = 1; j < samples; ++j) {
42      double d = *(double*)(in_array->data + j * in_array->strides[0] + c
            * in_array->strides[1]);
43      mav += fabs(d);
44
45      //WL
46      double pred = *(double*)(in_array->data + (j-1) *
            in_array->strides[0] + c * in_array->strides[1]);
47      wl += fabs(d - pred);
48      //ZC
49      double zcCoeff = pred*d;
50      zc += zcCoeff < 0.0 ? 1.0 : zcCoeff;
51      //ssc
52
53    }
54    //Finalization of feature for channel
55    mav = mav / (double)samples;
56    *(double*)(out_array->data + c * out_array->strides[0]) = mav;
57    *(double*)(out_array->data + (c + channel_stride) *
          out_array->strides[0]) = wl;
58    *(double*)(out_array->data + (c + channel_stride*2) *
          out_array->strides[0]) = zc;
59  }
60  for (int c = 0; c < channels; ++c) {
61    ssc = 0.0;
62
63    double d = *(double*)(in_array->data + 0 * in_array->strides[0] + c *
          in_array->strides[1]);
64    mav += fabs(d);
65
66    for (int j = 2; j < samples; ++j) {
67      double d = *(double*)(in_array->data + (j-1) * in_array->strides[0]
            + c * in_array->strides[1]);
68      mav += fabs(d);
69
```

```
70      double pred = *(double*)(in_array->data + (j-2) *
            in_array->strides[0] + c * in_array->strides[1]);
71      wl += fabs(d - pred);
72
73      double next = *(double*)(in_array->data + (j) *
            in_array->strides[0] + c * in_array->strides[1]);
74      wl += fabs(d - pred);
75
76      double sscCoef = (d - pred)*(d - next);
77      ssc += sscCoef >= 0.0 ? 1.0 : sscCoef;
78    }
79    *(double*)(out_array->data + (c + channel_stride * 3) *
          out_array->strides[0]) = ssc;
80  }
81
82  //Py_INCREF(out_array);
83  return (PyObject *)out_array;
84
85  /*  in case bad things happen */
86 fail:
87  Py_XDECREF(out_array);
88  return NULL;
89 }
```

Le caratteristiche di Hudgins vengono così calcolate in modo efficiente, utilizzando gli stessi tipi di dati che erano utilizzati nell'implementazione in Python.

6.4.2 Classificatore FNN

Il modulo *Classification* contiene la classe *MatlabNN*, la quale è stata implementata seguendo il modello di rete neurale specificato nel capitolo *Progettazione del sistema e fase di training con Matlab*.

Nonostante il codice sia interamente scritto in Python, le performance della funzione `update()` sono buone (verranno discusse quantitativamente nella sezione successiva); di conseguenza questa parte dell'algoritmo di classificazione non ha richiesto un raffinamento in linguaggio *C* come la funzione di estrazione delle caratteristiche.

6.5 Valutazione dei tempi di esecuzione

Per verificare che il dispositivo indossabile rispetti i requisiti per la classificazione in tempo reale sono stati effettuati dei test per calcolarne il tempo di esecuzione.

I test sono stati eseguiti utilizzando lo strumento *Line_profiler* che consente di analizzare linea per linea il codice, riportando il tempo di esecuzione e la percentuale di utilizzo.

Nella versione finale del software si è ottenuto un tempo di acquisizione totale medio di $13ms$, risultato ottenuto integrando i risultati dei vari test di profilazione; il *Wall Time* percepito dall'utente invece è stato di $16ms$. Tutti i tempi riportati sono stati raccolti sull'analisi di un singolo segmento da 500 campioni (equivalente a $500ms$ di segnale campionato a $1kHz$).

Durante l'esposizione dei tempi di esecuzione si farà riferimento solo al classificatore a due fasi con valutazione singola degli stati burst e steady.

I dati raccolti sono visibili in tabella 6.1 dove sono stati raggruppati in *Acquisizione*, *Classificazione*, *Overhead di Gestione*. Più avanti verranno presentati e commentati i dettagli della profilazione.

Operazione	Tempo (ms)
Acquisizione	2.4 ms
Classificazione	6.16 ms
Overhead	4.78 ms
TOTALE	13.34 ms

Tabella 6.1: Le tempistiche globali di esecuzione sono raggruppate per acquisizione (tempo necessario ad acquisire il segmento), tempo di classificazione (tempo necessario per produrre la variabile di controllo) e overhead (che indica tutti i costi di gestione e di comunicazione)

Nella prima versione del dispositivo si è ottenuto un *Wall Time* di $174ms$, principalmente dovuto alla funzione Hudgins non ottimizzata. L'uso delle *C API* ha consentito di classificare in tempo reale con qualsiasi spostamento della finestra di acquisizione compreso tra i $62ms$ e i $250ms$.

6.5.1 Tempistiche delle Funzioni di Classificazione

Di seguito sono state raccolte le tempistiche riguardanti la sola funzione di elaborazione dei segmenti.
Questa analisi particolareggiata della funzione di calcolo ci ha consentito di identificare i colli di bottiglia che sono poi stati ridotti fino ad arrivare alla versione che è stata presentata in questo lavoro.

Nella tabella 6.2 vengono presentati i tempi di profilazione dettagliati, raggruppati per passo di classificazione (*Estrazione delle Caratteristiche*, *Riduzione delle Caratteristiche e normalizzazione*, *Classificazione*). I dati sono

Operazione	Tempo (ms)
Hudgins	0.3 ms
PCA e Normalizzazione	1.4 ms
Classificazione FFNN50	2.9 ms
Totale Primo Stadio	4.6 ms
PCA e Normalizzazione	0.7 ms
Classificazione FFNN50	2.5 ms
Totale Secondo Stadio	3.1 ms
Overhead	0.1 ms
Stadio Singolo	4.6 ms
Stadio Combinato	7.8 ms
Tempo Medio	6.2 ms

Tabella 6.2: Le tempistiche in questa tabela riportano i tempi di esecuzione del classificatore a doppia fase, raggruppando i risultati prima per funzionalità, poi per stadio di classificazione ed infine viene riportato il tempo medio di classificazione, calcolato tenendo conto della frequenza di attivazione di ogni stadio.

poi raggruppati per stadio, facendo riferimento al classificatore a due fasi. Viene poi indicato il tempo totale dello stadio singolo, che divide lo stato di Rest dallo stato attivo del segnale e il tempo totale dello stadio combinato che somma il tempo di esecuzione del primo stadio con il secondo stadio responsabile della classificazione sulle otto classi di movimento.

Durante l'esecuzione su un tracciato EMG lo stadio singolo viene attivato con frequenza media del 53%; si può stimare il tempo medio di esecuzione della funzione di classificazione pari a $6.2ms$, dato che il secondo stadio viene attivato solo dove realmente necessario.

Nei momenti in cui l'attività muscolare ha frequenza maggiore del 53% registrato nei tracciati di prova, il tempo di calcolo tenderà ai $7.8ms$ mentre invece nei momenti di inattività il classificatore attiverà solo il primo stadio, abbassando il tempo di calcolo complessivo.

6.5.2 Tempistiche del meccanismo di gestione Multithreading

Si è deciso di valutare anche l'overhead dovuto al meccanismo di gestione multithreading. Facendo riferimento al listato 6.1, la tabella 6.3 espone i costi di gestione del meccanismo multithread.

Operazione	Temp (ms)	% sul totale
	_loopR	
Lettura	2.4 ms	78%
Gestione	0.68 ms	22&
	_loopP	
Classificazione	6.2 ms	60%
Gestione	4.1 ms	50%
Totale Gestione	4.78 ms	36%

Tabella 6.3: I tempi riportati sono stati divisi in classificazione e lettura, riportando per ognuno il costo di gestione dei thread.

Il costo di gestione corrisponde al 36% del tempo totale di esecuzione, ma questo meccanismo risulta indispensabile. Il motivo dell'indispensabilità di utilizzare due thread per svolgere i due compiti deve essere ricercato nelle modalità di funzionamento della seriale utilizzata.

Ammesso di utilizzare la frequenza di campionamento della scheda *Eracle* pari a $270Hz$, viene generato un campione ogni $3.7ms$ con un tempo di trasmissione di $0.2ms$ ad una velocità di 57600 $baud/s$ (ipotizzando 12 simboli trasmessi).

Ogni segmento prodotto dal nodo di input contiene 135 campioni, quindi il ritardo per ricevere un segmento di segnale EMG è pari all'apertura della finestra di $500ms$. Per ricevere le finestre successive, ipotizzando uno spostamento della finestra di acquisizione pari a 17 campioni, si ha un ritardo di $62ms$, mentre per una frequenza di acquisizione di $1kHz$ si registrano gli stessi ritardi ma con una numerosità della finestra pari a 500 campioni ed uno spostamento di 62 campioni. Durante questo periodo di inattività senza utilizzare un approccio multithreading, il calcolo rimarrebbe bloccato dall'acquisizione seriale, mentre così può essere analizzato il segmento acquisito in precedenza.

6.5.3 Consumi

I consumi dell'Intel Edison sono pari a $200mA$ durante l'esecuzione dell'algoritmo, mentre utilizzando il WiFi per monitorarne l'esecuzione il consumo è stato di $500mA$.

6.6 Performance di Classificazione

Questo lavoro si pone come obiettivo la progettazione e l'implementazione di un dispositivo indossabile capace di classificare il segnale elettromiografico nel suo complesso, sfruttando le informazioni comprese in tutto il tracciato e non solo nella fase steady o nella fase di burst. Per questo motivo è necessario anche affrontare lo studio del classificatore su una sequenza di segnale elettromiografico reale e confrontare i risultati con le performance ottenute durante l'apprendimento. Come prima cosa si valuterà il coefficiente di errore del classificatore sul tracciato EMG di test e, a fianco dell'analisi quantitativa, verrà anche discussa la qualità del tracciato di classificazione, valutando i margini di indecisione e la tipologia degli errori riscontrati.

Per questi test sono stati utilizzati due tracciati EMG di prova.
Il primo è stato utilizzato per testare il classificatore a fase singola *FFNN50*, il quale classifica solo su segnale steady. Il segnale artificiale è stato composto unendo dei segmenti di segnale appartenente a tutte le otto classi analizzate dalle quali sono state rimosse le parti di Rest e le parti di Burst.
Il secondo tracciato invece è composto da fasi complete di rest-burst-steady appartenenti alle otto classi di movimento più la classe di rest.
L'ultimo tracciato è equivalente ad una simulazione di reale utilizzo di una protesi, dove l'utente vuole attivare in successione i movimenti per effettuare una serie di compiti.

L'obiettivo del primo test è di mostrare come il classificatore si comporta in situazione di stress, quando il segmento è composto da segnale contenente dati provenienti da parti del segnale EMG molto diverse tra di loro.

Durante l'addestramento del classificatore, il segnale viene diviso in porzioni contenenti solo la caratteristica di interesse: il segmento della classe i-esima conterrà quindi solo le caratteristiche associate alla classe i-esima. In uno scenario reale però non è così, perché i segmenti vengono creati mano a mano che il segnale viene acquisito. Questo porta all'esistenza di segmenti contenenti segnale appartenente a *classi miste*, sui quali il classificatore non ha potuto apprendere. Si è voluto dimostrare come il classificatore si comporti bene anche in questi casi particolari, mantenendo delle performance stabili che consentano alla protesi di avere un funzionamento morbido.

Il secondo test punta a mettere in risalto il funzionamento dei classificatori a doppia fase su un segnale reale, nel quale i punti di criticità sono rappresentati dalla corretta divisione del segnale in segmenti Rest e in segmenti di movimento da parte del primo stadio di classificazione e dal comportamento del secondo stadio in presenza dei segmenti di burst.

Mentre per i segmenti di steady è stato ampiamente dimostrato in que-

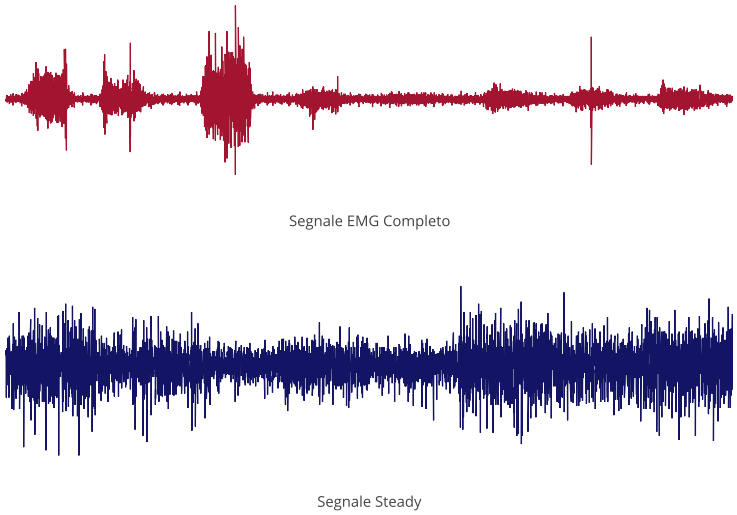


Figura 6.3: Tracciato EMG artificiale contenente solo le fasi steady.

sto lavoro come il classificatore *FFNN50* abbia delle performance stabili e superiori al *98.8%*, la classificazione del segnale di burst non si può ritenere stabile su tutta la lunghezza della fase di attivazione. Eńoto dallo studio di Cavazzana del 2011 [13] che la fase di burst non consente una classificazione stabile. Come è visibile in figura 6.4, il contenuto informativo durante questa fase varia partendo da una capacità di classificazione del *15%* fino ad una capacità superiore all'*80%* nella seconda metà della fase.

6.6.1 Test Classificazione FFNN a 50 neuroni sul segnale Steady

Il primo test dimostra la capacità del classificatore di mantenere le prestazioni di classificazione anche con segmenti contenenti segnale appartenente a classi miste. Il tracciato di prova è rappresentato in figura 6.3 ed è composto da solo segnale steady concatenato, per creare un netto cambio di classe.

Guardando solamente la quantità di segmenti classificati correttamente, il classificatore ha avuto delle performance dell'81%, nettamente peggiori rispetto a quanto riscontrato in fase di apprendimento.

In figura 6.5 è visibile il plot dei risultati ottenuti: nel primo grafico sono rappresentate le classi originali, mentre nel secondo è visibile il risultato della classificazione da parte della rete neurale (in Rosso gli errori e in Blu

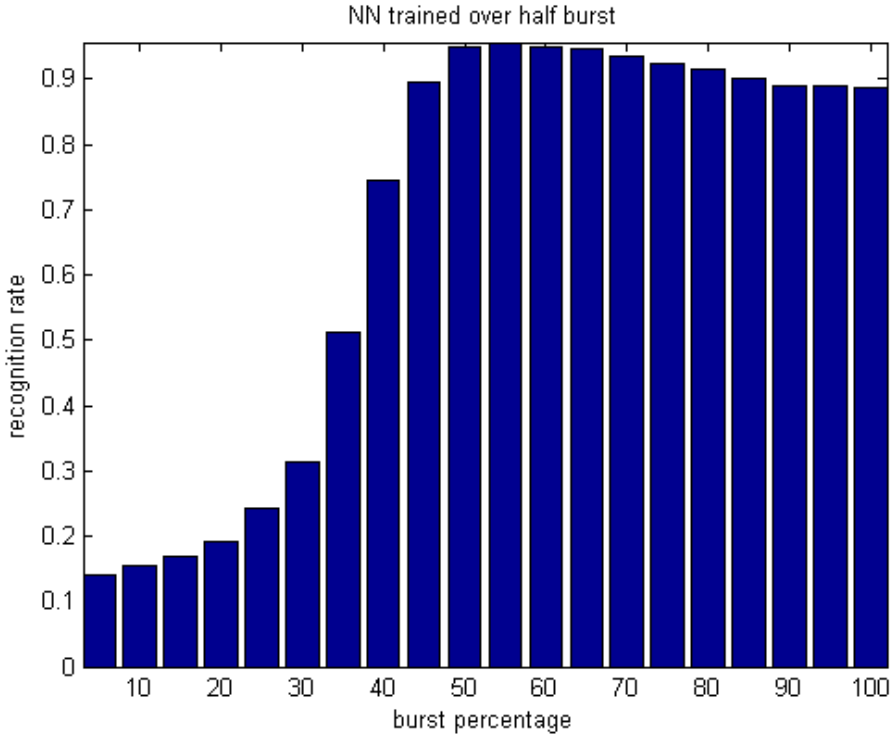


Figura 6.4: Variazione della capacità di classificazione lungo la fase di burst [13]

le etichette corrette); nell'ultimo grafico è rappresentato l'output numerico della rete neurale, indicativo dei margini di confidenza con cui l'algoritmo ha preso la decisione.

Analizzando con accuratezza la posizione degli errori di classificazione, si è scoperto che il classificatore continuava a classificare con performance del 98.8% all'interno del segnale steady, ma classificava erroneamente tutti i valori di frontiera. Questo comportamento è dovuto alla grandezza della finestra di segmentazione: maggiore è l'ampiezza della finestra maggiori saranno i segmenti con *classe mista* processati (con *classe mista* si intendono tutti quei campioni che contengono dati provenienti da una classe di movimento ma anche da un'altra). Analizzando anche il tracciato di classificazione delle rete neurale, dove è visibile l'intervallo di confidenza, si è visto che il sistema classifica i segmenti a classe mista con un'etichetta appartenente ad una delle due classi ed in particolare alla classe preponderante del segmento misto; questo aspetto è positivo perché dimostra come il calo delle performance non sia dovuto a delle cattive proprietà del classificatore ma solamente ad

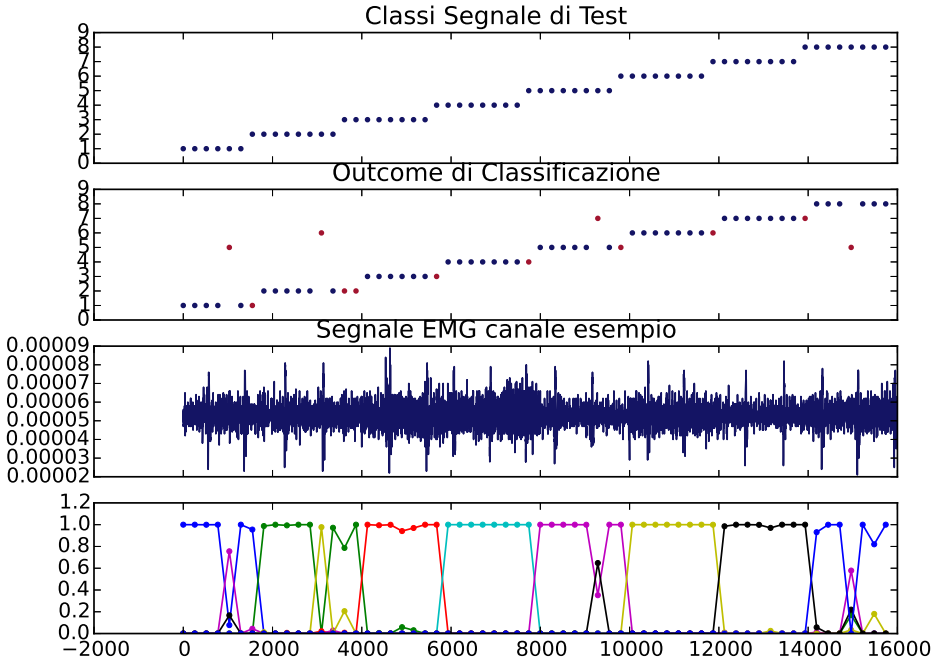


Figura 6.5: Performance di classificazione del classificatore FFNN50 sul tracciato contenente solo la fase steady. Nel primo grafico sono presenti le classi originali, nel secondo è presente l'output di classificazione (in rosso vengono indicati gli errori e in blu le etichette classificate correttamente), nel terzo viene mostrato il segnale EMG artificiale e nell'ultimo vengono indicati i margini di confidenza con cui l'algoritmo ha preso la decisione.

un ritardo nella percezione del cambio di classe, dovuto principalmente alla segmentazione. Questo ritardo può essere quantificato a $250ms$, utilizzando una finestra di 500 campioni con un segnale campionato a $1kHz$. Il valore del ritardo è stato stimato osservando attentamente una transizione, e il cambio di tendenza nella classificazione avveniva a metà segmento, quando cioè aumentava la presenza della classe successiva e quindi questa influenzava maggiormente le caratteristiche estratte.

Questo comportamento non è sintomo di errore ma di stabilità del classificatore implementato, dato che nella maggioranza dei casi quando ci si trova in presenza di una classe mista la scelta ricade su una delle due classi presenti nel segmento. Questa proprietà dipende dal set utilizzato per l'estrazione delle caratteristiche: Hudgins contiene delle caratteristiche integrative che

variano in modo continuo allo spostarsi della finestra di acquisizione, agendo come un filtro nei confronti dei bruschi cambi di classe.

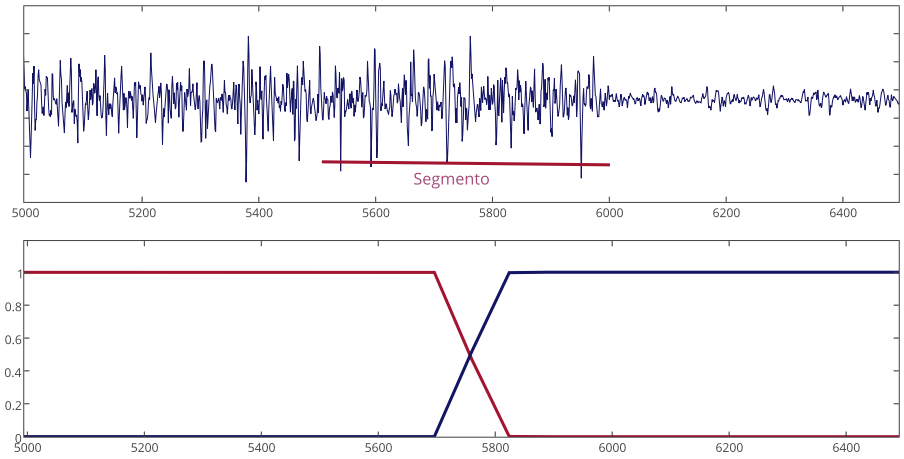


Figura 6.6: Dettaglio dell'andamento di classificazione sul segmento misto, mettendo in risalto come Hudgins consenta una classificazione morbida. L'asse orizzontale riporta i campioni, mentre l'asse verticale del secondo grafico rappresenta l'intervallo di confidenza relativo.

In figura 6.6 è visibile in dettaglio questo comportamento: mano a mano che la finestra procede verso la nuova classe la transizione tra le due è graduale e lineare, realizzando una trasformazione progressiva dalla classe uno alla classe due.

6.6.2 Test Classificatore a due Fasi con valutazione separata di steady e burst su un Tracciato EMG completo

Le performance teoriche del classificatore a due fasi con valutazione separata sono di 94% per quanto riguarda la classificazione del segnale di burst e di 96.4% per la fase di steady. Questo implica che tenendo conto dell'errore del primo stadio di classificazione il tasso di classificazione corretta è del 94% circa.

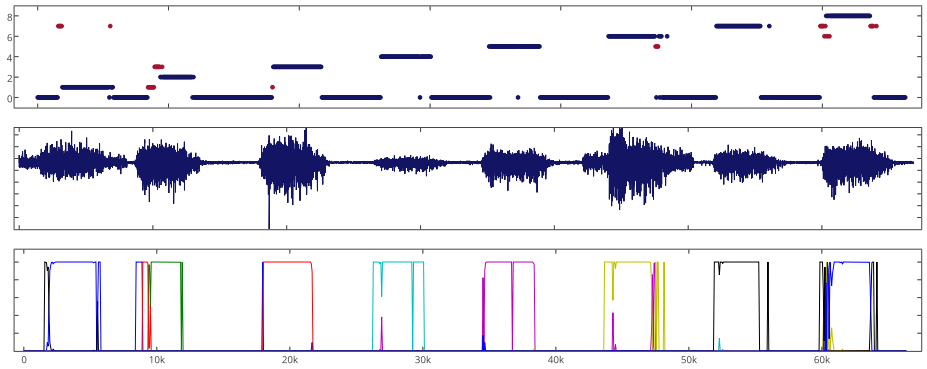


Figura 6.7: Test Classificatore a due Fasi con valutazione separata di steady e burst con percentuale di classificazione pari al 94.2%. Nel primo grafico viene mostrato l'output di classificazione (in rosso vengono indicati gli errori e in blu le etichette classificate correttamente), mentre nell'ultimo vengono indicati i margini di confidenza con cui l'algoritmo ha preso la decisione. L'asse orizzontale riporta i campioni, mentre l'asse verticale del secondo grafico rappresenta l'intervallo di confidenza relativo. Mentre nel primo grafico sull'asse orizzontale viene indicata l'indice della classe riconosciuta.

Nel tracciato di test eseguito sull'Intel Edison (visibile in figura 6.7) si sono ottenuti valori in linea con il risultato teorico (94%).Il classificatore con valutazione separata non solo ha mostrato un tasso di classificazione inferiore al classificatore a valutazione singola, ma presenta anche molto rumore in concomitanza della fase di burst, confermando come sia problematica da classificare correttamente per il suo contenuto variabile di informazione.

6.6.3 Test Classificatore a due Fasi con valutazione unita di steady e burst su un Tracciato EMG completo

Il classificatore a due fasi con valutazione unita ha migliorato le performance teoriche fino ad ottenere un tasso di classificazione del 98.6% sul tracciato di test. Osservando il grafico in figura 6.8 si può notare l'ottima stabilità nell'identificare la fase di rest. Anche in questo caso, come nel classificatore *FFNN50* semplice, la maggior parte degli errori sono localizzati nei punti

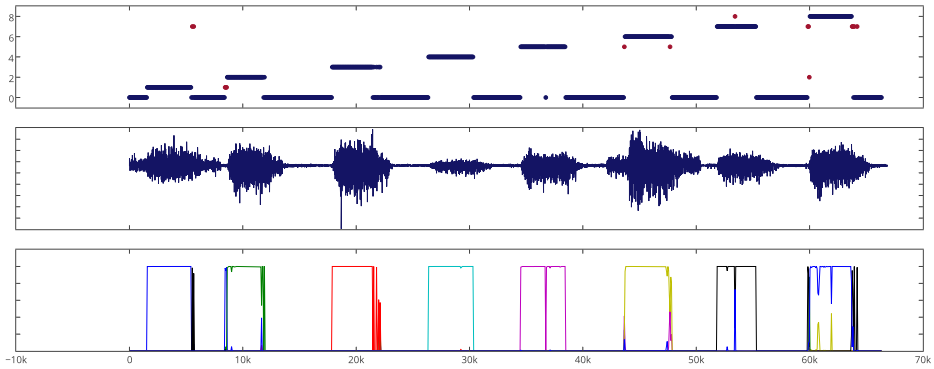


Figura 6.8: Test Classificatore a due Fasi con valutazione singola di steady e burst con percentuale di classificazione pari al 98.6%. Nel primo grafico viene mostrato l'output di classificazione (in rosso vengono indicati gli errori e in blu le etichette classificate correttamente), mentre nell'ultimo vengono indicati i margini di confidenza con cui l'algoritmo ha preso la decisione. L'asse orizzontale riporta i campioni, mentre l'asse verticale del secondo grafico rappresenta l'intervallo di confidenza relativo. Mentre nel primo grafico sull'asse orizzontale viene indicata l'indice della classe riconosciuta.

di confine tra due classi, e questo comportamento giustificabile è già stato osservato per il classificatore FFNN50, analizzato nelle sezioni precedenti.

Il risultato di questo nuovo approccio alla classificazione del segnale *EMG* rappresenta uno dei risultati più rilevanti di questo lavoro. Come già detto in precedenza, secondo lo studio di Hargrove classificando la fase di steady e la fase di burst contemporaneamente si ottiene un peggioramento delle performance di classificazione cercando di migliorare l'usabilità del sistema [5]. Con questo approccio la diminuzione delle performance non è stata così incidente da compromettere l'affidabilità del controllore, di conseguenza il nuovo classificatore aumenta le possibilità di applicazione rispetto a sistemi a singolo stadio, perché non ha bisogno di un altro sistema per eliminare la fase di rest e di burst prima di poter classificare il segnale elettromiografico.

7. Risultati

> Your time is limited, so don't waste it living someone else's life. Don't be trapped by dogma - which is living with the results of other people's thinking. Don't let the noise of others' opinions drown out your own inner voice. And most important, have the courage to follow your heart and intuition.
>
> Steve Jobs

Questo capitolo raccoglie i principali risultati ottenuti in questo lavoro di tesi, esponendo i dati più significativi. Si effettuerà un percorso organico attraverso i vari esperimenti effettuati nel corso dello sviluppo e della progettazione del sistema di classificazione, raggruppando per ogni aspetto chiave di questo lavoro i risultati intermedi, che sono andati a comporre il sistema nella sua completezza. Verranno fatti dei confronti con altri lavori per mostrare la bontà della scelte effettuate o evidenziare le peculiarità del sistema sviluppato;in particolare, vista la scarsa trattazione in letteratura, saranno fatti dei confronti anche tra zone muscolari diverse (spalla/mano), il che può essere utile per capire in quale direzione è possibile spostarsi per ottenere dei miglioramenti.

Numerosità dei canali

E' stato evidente fin dai primi lavori di Hudgins e Englehart [6] che la classificazione del segnale elettromiografico può essere affrontata efficacemente solo con un'acquisizione multicanale. D'altra parte un sistema di classificazione in tempo reale per un esoscheletro o una protesi deve poter essere poco costoso ed avere un peso ridotto. Questa problematica può essere risolta in parte con la riduzione dei canali utilizzati: durante la trattazione infatti si è partiti da un dataset composto da otto canali e si è cercato di evidenziare quelli a più alto contenuto informativo.

Valutando ognuno dei canali di acquisizione singolarmente, è stato proposto un ordinamento da quello che contribuisce maggiormente alla classificazione a quello con meno contenuto informativo. Tale ordine è stato proposto analizzando la variabilità dei valori assunti da ogni canale sia tra tutti i movimenti che per ogni movimento preso da solo, e successivamente è stato confermato analizzando le capacità di classificazione di ogni canale utilizzando l'intero dataset. L'ordine dei canali, con i corrispondenti muscoli, è così definito:

- Canale 1 - Grande Pettorale Clavicolare
- Canale 6 - Trapezio Ascendente
- Canale 3 - Serrato Anteriore
- Canale 7 - Infraspinato
- Canale 5 - Trapezio Trasverso
- Canale 4 - Trapezio Discendente
- Canale 8 - Gran Dorsale
- Canale 2 - Grande Pettorale Sternale

Tale ordinamento identifica come canale migliore il numero 1 e come canali peggiori il 2 e l'8. Questa classifica può essere utilizzata per comporre dei sottoinsiemi di canali, garantendo di ottenere la precisione di classificazione massima una volta scelto il numero di canali che si vuole utilizzare.

Considerando uno stadio di classificazione che unisca l'identificazione dei movimenti per il segnale steady e il segnale di burst e dimezzando il numero dei canali utilizzando il sottoinsieme $< 1, 3, 6, 7 >$, si hanno performance di accuratezza del 98.3%, mentre il tasso di classificazione si abbassa a 97.2% andando ad utilizzare solo tre canali nel sottoinsieme $< 1, 3, 6 >$.

Caratteristiche di Hudgins

Oltre agli ottimi risultati ottenuti riducendo il numero di canali, lo studio di comparazione effettuato tra le caratteristiche di Phyniomark e le caratteristiche di Hudgins ha dimostrato la dominanza delle seconde per la classificazione real time. Come si è visto durante il test comparativo al capitolo *Progettazione del sistema e fase di training con Matlab* alla sezione 5.2.3, queste caratteristiche accoppiate con un classificatore a rete neurale consentono di ottenere risultati paragonabili a quelle del set di Phyniomark (99.8%), impiegando tuttavia un tempo decisamente minore in fase di calcolo. L'insieme di Hudgins non solo si è imposto per la velocità di calcolo, ma ha anche dimostrato di essere meno sensibile alla variazione dell'apertura della finestra di classificazione. Generalmente maggiore è la finestra di classificazione maggiore è il contenuto informativo raccolto per la classificazione, ma le caratteristiche di Hudgins hanno dimostrato di avere una variazione minima delle performance dimezzando la finestra di acquisizione; questo risultato è decisamente importante, dato che una reattività maggiore della protesi o dell'esoscheletro richiede necessariamente una finestra di acquisizione minore.

Questo insieme di caratteristiche ha dimostrato di possedere un'ottima robustezza durante i test sul dispositivo indossabile; dato che, per la natura particolare delle caratteristiche inserite in questo insieme, la presenza di perturbazioni o bruschi cambi di tendenza all'interno del tracciato elettromiografico producono un risultato di classificazione dalle transizioni morbide, capace di filtrare questo tipo di interferenze.

I classificatori a due fasi

Per migliorare il funzionamento del dispositivo durante la classificazione online è stato proposto un classificatore a più fasi, il cui scopo è di ridurre la complessità della singola classificazione separando i gruppi di informazioni indipendenti per ottenere un migliore risultato dei singoli classificatori.

Sono state proposte due tipologie di divisione e in entrambe si è voluto separare il problema della identificazione della fase attiva dalla fase di riposo. Nella *prima fase* ci si occupa di decidere se il segmento che si sta osservando appartiene ad una fase di Rest oppure a una fase attiva e solo in questo ultimo caso si invoca il secondo classificatore che dovrà classificare il tipo di movimento presente nel segmento. Questa scelta di differenziare e specializzare i classificatori consente di ridurre l'errore durante la classificazione online.

I due tipi di classificatore si differenziano per la modalità di classificazione del segnale attivo (*seconda fase* del classificatore): nella prima tipologia

la fase di steady e la fase di burst sono classificate separatamente, mentre nella seconda tutto il segnale attivo viene trattato come appartenente a un unica fase combinata. Il primo approccio porta ad una classificazione totale peggiore rispetto al secondo a causa dell'alta concentrazione di rumore di classificazione durante la fase di burst, mentre il secondo approccio con valutazione singola della fase steady/burst consente di avere ottime prestazioni (tasso di classificazione del 98%).

Prestazioni del dispositivo indossabile

Un utilizzo in tempo reale richiede una buona reattività del dispositivo mioelettrico, per consentire un utilizzo fluido al suo utilizzatore. In un dispositivo completo molti sono i fattori che concorrono al tempo di reazione totale, in questo studio ci si è occupati principalmente dei tempi di classificazione delegando un'analisi dei tempi di acquisizione, di controllo e di attuazione della protesi o esoscheletro a studi futuri. Cruciali sono stati i risultati ottenuti per quanto riguarda le tempistiche di classificazione: il dispositivo implementato ha un tempo massimo di attesa dall'acquisizione dei campioni di segnale elettromiografico, fino alla produzione della scelta di classificazione, di $17ms$, il che comporta alla possibilità di classificare ad una frequenza di $59Hz$. Questo risultato è stato ottenuto prendendo in considerazione una finestra di acquisizione di $500ms$ e uno spostamento della finestra di acquisizione (offset) di $62ms$ (a $1KHz$). Il dispositivo si trova ad analizzare 17 nuovi segmenti ogni secondo, avendo la capacità di processarne 59.

Comparazione con altri lavori presenti in letteratura

Per paragonare il risultato ottenuto si fa riferimento ad uno studio di tesi del Politecnico di Milano [13], dove viene sviluppato un dispositivo per l'identificazione del segnale elettromiografico di arto superiore ma relativo ai movimenti della mano, prelevando il segnale a livello dell'avambraccio.

Nel lavoro di Cavazzana, che risale al 2011, si era ottenuto un tempo di classificazione di $35ms$, acquisendo tre canali con una finestra di 100 campioni, per una frequenza di classificazione di 28 analisi al secondo; un risultato doppio rispetto a quello ottenuto in questo sviluppo di tesi ($17ms$).

Il nuovo dataset acquisito con Eracle

Visti i risultati promettenti ottenuti con il dataset su cui è stato effettuato lo studio e le buone prestazioni mantenute anche con l'utilizzo di soli tre canali, si è deciso di testare il percorso di classificazione proposto anche su un dataset acquisito dalla scheda Eracle utilizzando i seguenti muscoli:

- Grande Pettorale Clavicolare (canale 1 in tabella 4.1)
- Serrato Anteriore (canale 3 in tabella 4.1)
- Trapezio Ascendente (canale 6 in in tabella 4.1)

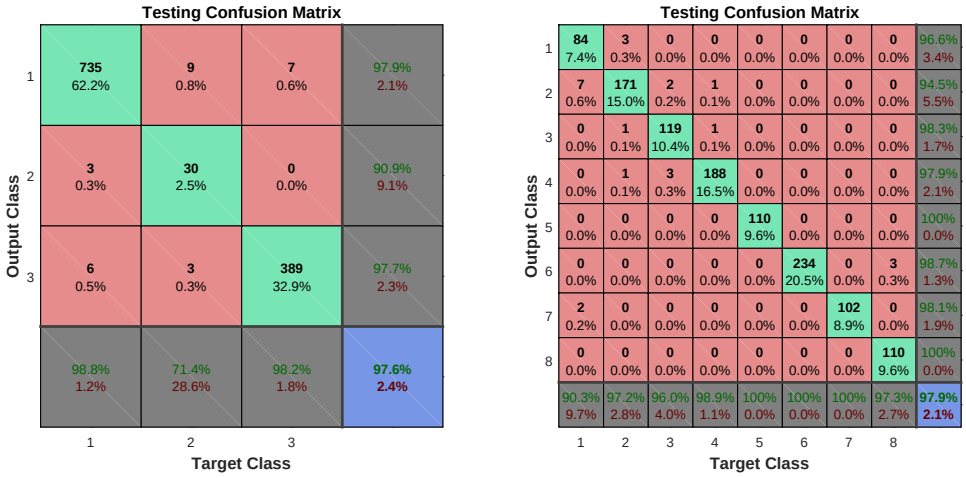
L'acquisizione è stata poi analizzata utilizzando lo stesso percorso di classificazione utilizzato per il dataset di Rivela e Scannella a 8 canali, modificando i parametri riguardanti la frequenza di acquisizione (apertura della finestra pari a 135 campioni, con spostamento di 17 campioni).
Analizzando le acquisizioni si è notato che era presente cross-talking tra i canali e il segnale rilevato dai muscoli era molto più debole. Questa tipologia di disturbi sono da ricercarsi nelle caratteristiche morfologiche della spalla dove sono presenti muscoli molto vicini tra loro e poco superficiali; si è riscontrato anche un notevole rumore dovuto al battito cardiaco, rilevato prevalentemente sul grande pettorale. Con l'applicazione dell'analisi delle componenti indipendenti (descritta al capitolo *Pattern Recognition per il segnale EMG* alla sezione 2.1.1 corrispondente) sarebbe stato possibile ottenere un miglioramento del segnale, ma lo scopo di questo nuovo dataset era quello di valutare il percorso elaborato in una situazione limitata sia dal numero di canali che dal rumore.

Nonostante queste limitazioni, con la capacità di riconoscere il segnale attivo pari a 97.6% (prima fase) e di classificarlo correttamente con una probabilità del 98% (seconda fase), si è riusciti a classificare correttamente il segnale proveniente dalla scheda *Eracle* il 95% delle volte. I dati di classificazione completi sono visibili nel dettaglio nella figura 7.1.

Applicazione in altre parti di arto superiore

Come prova ulteriore della superiorità delle Caratteristiche di Hudgins unite alle reti neurali, per l'estrazione dell'intenzione di movimento dal segnale elettromiografico, si sono voluti presentare anche i risultati ottenuti su un altro dataset raccolto per mezzo della scheda di acquisizione *Eracle* prelevato in uno studio di tesi precedente svolto da Luca Cavazzana nel 2011 [13]. Questo test aggiuntivo si è svolto su un segnale prelevato dalla mano ed ha lo scopo di mostrare la capacità del percorso di identificazione proposto ad adattarsi a nuove situazioni, mettendo in risalto le performance di classificazione su una zona del corpo diversa da quella per cui è stato inizialmente progettato (dalla spalla alla mano).

Con il dataset riguardante la mano vengono identificati correttamente i movimenti sull'intero segnale elettromiografico, ottenendo un indice di classificazione corretta pari al 93%. Il dominio applicativo sicuramente richiede



(a) **(b)**

Figura 7.1: Matrici di confusione per il dataset raccolto con eracle **(a)** Prestazioni sulla divisione dei segmenti in Rest, Burst e Steady (accuratezza del 97.6%). **(b)** Prestazioni di classificazione degli otto movimenti (Burst + Steady) (accuratezza del 97.9%)

dei raffinamenti dell'algoritmo per meglio adattarsi alla situazione diversa, ma dimostra sicuramente la generalità dell'approccio proposto in questa tesi.

8. Conclusioni

"Begin at the beginning," the King said, very gravely, "and go on till you come to the end: then stop."

Lewis Carroll, Alice in Wonderland

L'obiettivo principale di questo lavoro di tesi è stato lo sviluppo e l'implementazione di un dispositivo di classificazione elettromiografica di superficie che fosse adatto alla realizzazione di una protesi o di un esoscheletro riabilitativo.
La principale differenza tra i lavori presenti in letteratura e ciò che si è ottenuto in questo studio di tesi è la finalità per cui il dispositivo è stato progettato; sono molti gli studi che trattano l'identificazione a scopo di analisi mentre la finalità di questa tesi è di proporre delle metodologie che possano essere integrate con successo su un dispositivo mioelettrico completo.

Per raggiungere questo obiettivo è stato necessario analizzare ogni aspetto relativo alla classificazione del segnale sEMG, partendo dalla scelta ottimale di una finestra di classificazione, identificando poi il migliore insieme delle caratteristiche e il meccanismo ideale per la sua riduzione, ricercando un classificatore che garantisse un funzionamento corretto anche in situazioni critiche.
Oltre a definire un buon percorso di classificazione (*Segmentazione*, *Estrazione delle Caratteristiche*, *Riduzione delle Caratteristiche*, *Classificazione*), ci si è anche occupati di aspetti trasversali alle sole prestazioni di classificazione, prima tra tutte l'usabilità di una futura protesi o esoscheletro mioelettrico.

Per garantire questo aspetto sono stati rispettati i vincoli di calcolo in tempo reale ed è stato definito un meccanismo efficace per la riduzione dei canali necessari alla classificazione, consentendo di mantenere ottime prestazioni anche con meno della metà dei canali originari di acquisizione. La diminuzione del numero di canali, anche se non risulta un requisito evidente come la necessità di un tempo di risposta inferiore, è comunque un aspetto più sottile ma che concorre in modo sostanziale alla buona riuscita di un dispositivo usabile. Infatti comporta un risparmio dal punto di vista energetico richiedendo minori canali di acquisizione e un hardware più semplice, il quale consente anche di abbassare i costi di produzione e sviluppo; inoltre consente la costruzione di una dispositivo mioelettrico più leggero e dalle dimensioni contenute(esoscheletro o protesi che sia).

La base di partenza per l'elaborazione di questa tesi è lo studio di Rivela e Scannella [1], le quali si sono occupate di classificazione dei movimenti dei muscoli della spalla, ma con un'ottica improntata all'analisi delle metodologie più che ad uno sviluppo di un dispositivo reale.
L'evolversi dell'attuale lavoro è avvenuto in una direzione differente, con lo scopo di renderlo attuabile nella realtà. Questo ha fatto sì che venissero confermati alcuni risultati ma al tempo stesso ridiscutendone altri a causa della diversa destinazione di impiego.

Di seguito verrà data risposta agli obiettivi prefissati, descrivendone le conclusioni ottenute.

Vincolo di esecuzione in tempo reale

É opinione diffusa che un tempo di risposta superiore ai 300 ms sia inaccettabile per un utilizzo fluido del dispositivo finale, perché ciò non darebbe la sensazione di naturalezza del movimento. Risulta perciò necessario mantenere il tempo totale di acquisizione al di sotto di questo valore limite.
Il dispositivo proposto è capace di fornire una variabile di controllo ogni $17ms$ dopo aver acquisito il segnale quindi con una frequenza superiore alla frequenza con cui i segmenti vengono acquisiti; inoltre il meccanismo multi-threading implementato consente di analizzare un segmento sEMG mentre viene acquisito il successivo, consentendo di processare il segnale senza aggiungere sostanziali ritardi rispetto allo spostamento della finestra di acquisizione (pari a $62ms$).

Dispositivo indossabile

La scelta del dispositivo Edison di Intel ha consentito di mantenere i consumi ridotti avendo le capacità di calcolo di un piccolo computer racchiuse nello

stesso spazio di una scheda *SD*.

Prestazioni Statiche

Durante l'utilizzo di una protesi è importante che una volta identificato il movimento la classificazione continui in modo stabile senza rumore nelle variabili di controllo. Il classificatore a rete neurale proposto, assieme alle caratteristiche di Hudgins, è riuscito ad ottenere prestazioni di 99.8% sul segnale steady, migliori rispetto a quanto riportato in letteratura (97.4%).

Performance dinamiche

Oltre all'identificazione delle posizioni raggiunte analizzando il segnale steady, anche l'analisi del segnale nella sua completezza è un requisito fondamentale per un dispositivo elettromiografico: è stato ideato quindi un approccio innovativo in ambito elettromiografico. L'approccio consiste nell'utilizzare diversi classificatori per identificare le diverse fasi del segnale, in modo tale da avere un risultato localmente migliore rispetto ad un classificatore generale che agisce tenendo conto di tutte le possibili combinazioni di uscita.
Grazie a questo tipo di classificatore, denominato generalmente nella tesi *classificatore a due fasi*, si è riusciti ad ottenere il 98% di precisione sulla classificazione delle otto classi di movimento più il segnale Rest, utilizzando il segnale elettromiografico nella sua completezza.

Riduzione del numero di canali

Anche se l'analisi multicanale è ritenuta la strategia migliore per l'analisi e l'identificazione dei movimenti basati sul segnale elettromiografico, non sempre utilizzare un grande numero di elettrodi è la migliore scelta, specialmente se si tratta di costruire una protesi dai consumi ridotti, economica e leggera. Proprio per questo motivo è stato condotto uno studio volto alla riduzione del numero di canali necessari a classificare con efficacia il segnale elettromiografico. Come risultato si è riusciti a ridurre i canali necessari da otto a tre mantenendo le prestazioni di classificazione al di sopra del 97%, mentre dimezzando il numero di canali le performance sono state per lo più invariate.

8.1 Sviluppi Futuri

Nonostante i risultati siano promettenti ci sono ancora delle questioni che dovrebbero essere approfondite, e in questa sezione vengono proposti alcuni sviluppi che possono essere affrontati in studi futuri.

Applicazione reale in una protesi o in un esoscheletro

Lo sviluppo più ovvio risulta essere quello di applicare su un dispositivo fisico mioelettrico la fase di classificazione progettata in questa tesi, definendo quindi anche la fase di controllo del metodo generale della Pattern Recognition e sviluppando uno strumento in grado di effettuare meccanicamente il movimento prescelto dal classificatore.
Il sistema ideato in questa tesi può essere utilizzato sia per una protesi che, ad esempio, per un esoscheletro in caso di deficit motori.

Acquisizione del segnale elettromiografico

Nonostante i risultati siano stati ottimi sia durante le prove effettuate sul segnale acquisito con il sistema professionale *BTS* sia con il segnale acquisito dalla scheda *Eracle*, la progettazione di una scheda di acquisizione elettromiografica a basso costo adatta ad un utilizzo su un dispositivo elettromiografico multicanale è un buon punto di espansione per il lavoro corrente.
Difatti ad oggi non esiste un vero e proprio strumento di acquisizione indossabile, a costi contenuti, che abbia una precisione tale da poter essere paragonato a un dispositivo professionale e che possa essere usato in maniera personale.
Ciò potrebbe essere applicato ad esempio nel campo del fitness, per la valutazione di parametri fisici. In un'era in cui si cerca sempre più di introdurre la tecnologia nella vita quotidiana delle persone, in ambito sportivo si potrebbe migliorare l'informazione dei muscoli che vengono maggiormente usati durante l'allenamento e se tali muscoli lavorano correttamente oppure no.
Un'altra applicazione di un dispositivo di acquisizione indossabile è l'ausilio per il training di protesi o esoscheletro; infatti si sta cercando sempre più di creare degli strumenti in grado di rendere autonoma anche la fase di training per il paziente, in modo tale da poterla svolgere anche comodamente a casa propria. Questa scelta è dettata dal fatto che un ambiente familiare e una certa indipendenza possono facilitare la pratica nel paziente, e anche perché ciò diminuirebbe i costi sanitari di assistenza.

Trattandosi di una classificazione multicanale, spesso per movimenti di zone critiche del corpo dal punto di vista del rumore, si rende necessario un meccanismo di riduzione del cross-talking tra i muscoli integrato nella scheda di acquisizione. Come analizzato in maniera approfondita nella tesi di Cavazzana [13], una delle metodologie più efficaci è l'ICA (Analisi delle Componendi Indipendenti) la quale, se ha sufficienti canali di acquisizione, riesce a limitare il cross-talking dovuto alla combinazione dei segnali dei diversi canali, separandone le sorgenti originali. Uno degli aspetti più impegnativi

dell'applicazione di questa tecnica all'acquisizione del segnale elettromiografico è rappresentata dall'impossibilità di utilizzare un alto numero di canali di acquisizione rispetto al numero di sorgenti originali. Una soluzione a questo problema è tuttavia stata presentata nella tesi sopracitata [13].

La scheda di acquisizione *Eracle* utilizza una frequenza di acquisizione appena sufficiente a percepire tutta la lunghezza spettrale del segnale elettromiografico, per questo in una progettazione di una nuova scheda di acquisizione potrebbe essere interessante aumentare la frequenza di acquisizione allineandola ai sistemi professionali ($1kHz$).

Ottimizzazione Hardware

Un'altra possibile direzione di espansione è quella di ridurre ulteriormente i tempi di esecuzione passando da un'implementazione software ad un'implementazione completamente hardware su FPGA dell'algoritmo di classificazione. Questa strategia è particolarmente indicata per il classificatore proposto, in quanto sia le caratteristiche di Hudgins, sia la classificazione per mezzo delle reti neurali possono essere parallelizzati per abbassare ulteriormente il tempo di classificazione e spostarsi dalle analisi in tempo reale alle analisi ad alta frequenza, particolarmente utili nel campo della predizione e del controllo.
Ovviamente questo percorso è consigliato solo in ambito industriale, una volta che il codice è stato ben testato e non sono necessarie modifiche frequenti al software, al fine di ridurre la riprogrammazione dell'FPGA e mantenere quindi prestazioni elevate.

Identificazione stati di transizione

Durante le fasi preliminari che hanno portato alla scrittura di questa tesi, sono stati condotti degli esperimenti sull'uso delle *SOFM (Mappe di Feature a Costruzione Autonoma)*. I test volevano dimostrare la fattibilità di una classificazione in tempo reale basata sulle transizioni effettuate dall'utilizzatore della protesi, piuttosto che sull'anticipazione. Questo tipo di identificazione senza supervisione classifica il segnale elettromiografico durante l'uso costruendo dei gruppi di nodi che si attivano quando viene eseguito il movimento di transizione tra una classe di movimento ad un'altra. Questo approccio non è stato ampliato nel corso di questa tesi poiché i risultati non erano all'altezza dell'obiettivo che si voleva raggiungere, ma valutando nuovamente questo metodo con la dovuta attenzione può sicuramente portare a risultati degni di nota.

A. Riferimenti ai codici Matlab e Python

L'intero codice Matlab è presente all'interno del seguente repository:

`https://bitbucket.org/MazLiz/thesiemg-matlabcode.git`

Il codice Python, utilizzato per l'implementazione del dispoditivo Intel Edison, si trova invece al seguente repository:

`https://bitbucket.org/thesiemg/edisoncode`

Bibliografia

[1] D. Rivela and A. Scannella, "Classificazione del segnale sEMG tramite pattern," Master's thesis, Politecnico di Milano, 2013.

[2] S. Watanabe, *Pattern Recognition: Human and Mechanical*. New York, NY, USA: John Wiley & Sons, Inc., 1985.

[3] A. K. Jain, R. P. W. Duin, and J. Mao, "Statistical pattern recognition: A review," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, pp. 4–37, Jan 2000.

[4] L. Smith and L. Hargrove, "Comparison of surface and intramuscular EMG pattern recognition for simultaneous wrist/hand motion classification," *Engineering in Medicine and Biology Society (EMBC), 2013 35th Annual International Conference of the IEEE*, pp. 4223–4226, Jul 2013.

[5] L. Hargrove, K. Englehart, and B. Hudgins, "A comparison of surface and intramuscular myoelectric signal classification," *Biomedical Engineering, IEEE Transactions on*, vol. 54, pp. 847–853, May 2007.

[6] K. Englehart, B. Hudgin, and P. Parker, "A wavelet-based continuous classification scheme for multifunction myoelectric control," *Biomedical Engineering, IEEE Transactions on*, vol. 48, pp. 302–311, Mar 2001.

[7] E. Gruppioni, M. Chiossi, M. Troncossi, A. Cutti, A. Davalli, and V. Parenti-Castelli, "A new active shoulder prosthesis: From the design to the first clinical application," *Myoelectric Symposium*, 2008.

[8] T. Kuiken, G. Li, R. Lock, B.A.and Lipschutz, L. Miller, K. Stubblefield, and K. Englehart, "Targeted muscle reinnervation for real-time myoelectric control of multifunction artificial arms," *Biomedical Engineering, IEEE Transactions on*, Feb 2009.

[9] A. Ajiboye and R. Weir, "A heuristic fuzzy logic approach to EMG pattern recognition for multifunctional prosthesis control," *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, vol. 13, pp. 280–291, Sep 2005.

[10] F. Sebelius, L. Eriksson, H. Holmberg, A. Levinsson, G. Lundborg, N. Danielsen, J. Schouenborg, C. Balkenius, T. Laurell, and L. Montelius, "Classification of motor commands using a modified self-organising feature map," *Med Eng Phys*, vol. 27, pp. 403–413, Jun 2005.

[11] G. Li, A. Schultz, and T. Kuiken, "Quantifying pattern recognition x2014;based myoelectric control of multifunctional transradial prostheses," *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, vol. 18, pp. 185–192, Apr 2010.

[12] G. Li, "Electromyography pattern-recognition-based control of powered multifunctional upper-limb prostheses, advances in applied electromyography," *ISBN: 978-953-307-382-8, InTech, DOI: 10*, p. 5772/22876., 2011.

[13] L. Cavazzana, "Integrating an EMG signal classifier and a hand rehabilitation device: Early signal recognition and real-time performances," Master's thesis, Politecnico di Milano, 2011.

[14] R. H. Chowdhury, M. B. Reaz, M. A. Ali, A. A. Bakar, K. Chellappan, and T. G. Chang, "Surface electromyography signal processing and classification techniques," *Sensors (Basel)*, vol. 13, no. 9, pp. 12431–12466, 2013.

[15] K. Englehart and B. Hudgins, "A robust, real-time control scheme for multifunction myoelectric control," *Biomedical Engineering, IEEE Transactions on*, vol. 50, pp. 848–854, Jul 2003.

[16] K. Englehart, *Signal Representation for Classification of the Transient Myoelectric Signal*. PhD thesis, Oct 1998.

[17] D. Farina and R. Merletti, "Comparison of algorithms for estimation of EMG variables during voluntary isometric contractions," *J Electromyogr Kinesiol*, vol. 10, pp. 337–349, Oct 2000.

[18] L. H. Smith, L. J. Hargrove, B. A. Lock, and T. A. Kuiken, "Determining the optimal window length for pattern recognition-based myoelectric control: balancing the competing effects of classification error

and controller delay," *IEEE Trans Neural Syst Rehabil Eng*, vol. 19, pp. 186–192, Apr 2011.

[19] A. Phinyomark, F. Quaine, S. Charbonnier, C. Serviere, F. Tarpin-Bernard, and Y. Laurillau, "EMG feature evaluation for improving myoelectric pattern recognition robustness," *Expert Systems with Applications*, vol. 40, no. 12, pp. 4832–4840, 2013.

[20] M. Pla Mobarak, R. Munoz Guerrero, J. Gutierrez Salgado, and V. Dorr, "Hand movement classification using transient state analysis of surface multichannel EMG signal," in *Health Care Exchanges (PAHCE), 2014 Pan American*, pp. 1–6, Apr 2014.

[21] B. Hudgins, P. Parker, and R. N. Scott, "A new strategy for multifunction myoelectric control," *IEEE Trans Biomed Eng*, vol. 40, pp. 82–94, Jan 1993.

[22] D. Yang, J. Zhao, L. Jiang, and H. Liu, "Dynamic hand motion recognition based on transient and steady-state EMG signals," *International Journal of Humanoid Robotics*, vol. 09, no. 01, p. 1250007, 2012.

[23] A. Phinyomark, C. Limsakul, and P. Phukpattaranont, "A novel feature extraction for robust EMG pattern recognition," *Journal of Computing 1*, vol. 1, pp. 71–80, Dec 2009.

[24] R. Boostani and M. H. Moradi, "Evaluation of the forearm EMG signal features for the control of a prosthetic hand," *Physiol Meas*, vol. 24, pp. 309–319, May 2003.

[25] E. Scheme and K. Englehart, "On the robustness of EMG features for pattern recognition based myoelectric control; A multi-dataset comparison," *Conf Proc IEEE Eng Med Biol Soc*, pp. 650–653, Aug 2014.

[26] M. Zardoshti-Kermani, B. Wheeler, K. Badie, and R. Hashemi, "EMG feature evaluation for movement control of upper extremity prostheses," *Rehabilitation Engineering, IEEE Transactions on*, vol. 3, pp. 324–333, Dec 1995.

[27] D. Tkach, H. Huang, and T. A. Kuiken, "Research study of stability of time-domain features for electromyographic pattern recognition," *J Neuroeng Rehabil*, vol. 7, p. 21, 2010.

[28] M. Lei, Z. Wang, and Z. Feng, "Detecting nonlinearity of action surface EMG signal," *Physics Letters A*, vol. 290, no. 5–6, pp. 297–303, 2001.

[29] A. Attenberger and K. Buchenrieder, "A matlab toolbox for upper limb movement classification," in *Computer Aided Systems Theory - EUROCAST 2013* (R. Moreno-Díaz, F. Pichler, and A. Quesada-Arencibia, eds.), vol. 8112 of *Lecture Notes in Computer Science*, pp. 191–198, Springer Berlin Heidelberg, 2013.

[30] A. Phinyomark, P. Phukpattaranont, and C. Limsakul, "Feature reduction and selection for EMG signal classification," *Expert Systems with Applications*, vol. 39, no. 8, pp. 7420–7431, 2012.

[31] O. Paiss and G. Inbar, "Autoregressive modeling of surface EMG and its spectrum with application to fatigue," *Biomedical Engineering, IEEE Transactions on*, vol. BME-34, pp. 761–770, Oct 1987.

[32] K. Englehart, B. Hudgins, P. A. Parker, and M. Stevenson, "Classification of the myoelectric signal using time-frequency based representations," *Med Eng Phys*, vol. 21, no. 6-7, pp. 431–438, 1999.

[33] I. Daubechies, *Ten Lectures on Wavelets*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 1992.

[34] A. Phinyomark, A. Nuidod, P. Phukpattaranont, and C. Limsakul, "Feature extraction and reduction of Wavelet Transform coefficients for EMG pattern classification," *Elektronika ir Elektrotechnika*, vol. 122, no. 6, 2012.

[35] X. Hu, Z. Wang, and X. Ren, "Classification of surface EMG signal using relative wavelet packet energy," *Comput Methods Programs Biomed*, vol. 79, pp. 189–195, Sep 2005.

[36] G. Wang, Z. Wang, W. Chen, and J. Zhuang, "Classification of surface EMG signals using optimal wavelet packet method based on davies-bouldin criterion," *Med Biol Eng Comput*, vol. 44, pp. 865–872, Oct 2006.

[37] M. Hussain, M. Reaz, F. Mohd-Yasin, and M. Ibrahimy, "Electromyography signal analysis using wavelet transform and higher order statistics to determine muscle contraction," *Expert Systems*, vol. 26, no. 1, pp. 35–48, 2009.

[38] A. Phinyomark, C. Limsakul, and P. Phukpattaranont, "Application of Wavelet Analysis in EMG Feature Extraction for Pattern Classification," *Measurement Science Review*, vol. 11, pp. 45–52, Jan 2011.

[39] N. Saito and R. Coifman, "Local discriminant bases and their applications," *Journal of Mathematical Imaging and Vision*, vol. 5, no. 4, pp. 337–358, 1995.

[40] L. E. Kavraki, "Dimensionality reduction methods for molecular motion," *OpenStax CNX*, vol. 12, 2007.

[41] J. Kruskal, "Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis," *Psychometrika*, vol. 29, no. 1, pp. 1–27, 1964.

[42] T. Kohonen, *Self-Organizing Maps*. Springer Series in Information Sciences 30, Springer-Verlag Berlin Heidelberg, 3 ed., 2001.

[43] J. U. Chu, I. Moon, and M. S. Mun, "A real-time EMG pattern recognition system based on linear-nonlinear feature projection for a multifunction myoelectric hand," *IEEE Trans Biomed Eng*, vol. 53, pp. 2232–2239, Nov 2006.

[44] S. Theodoridis and K. Koutroumbas, *Pattern Recognition*. Academic Press, 1999.

[45] K. Fu, *Syntactic pattern recognition and applications*. Prentice-Hall advanced reference series: Computer science, Prentice-Hall, 1982.

[46] K. S. Fu, "A step towards unification of syntactic and statistical pattern recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 3, pp. 398–404, 1986.

[47] L. Perlovsky, "Conundrum of combinatorial complexity," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 20, pp. 666–670, Jun 1998.

[48] A. Jain, J. Mao, and K. Mohiuddin, "Artificial neural networks: a tutorial," *Computer*, vol. 29, pp. 31–44, Mar 1996.

[49] K. Veer, "Experimental study and characterization of sEMG signals for upper limbs," *Fluctuation and Noise Letters*, vol. 14, no. 03, p. 1550028, 2015.

[50] K. Muller, C. Anderson, and G. Birch, "Linear and nonlinear methods for brain-computer interfaces," *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, vol. 11, pp. 165–169, Jun 2003.

[51] T. Poggio and F. Girosi, "Regularization algorithms for learning that are equivalent to a multilayer networks," *Science*, vol. 247, pp. 978–982, Feb 1990.

[52] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference and Prediction*. Springer, 2001.

[53] P. Belluco, M. Bordegoni, and U. Cugini, "Eracle: Electromyography system for gesture interaction," in *Human Interface and the Management of Information. Interacting with Information* (M. Smith and G. Salvendy, eds.), vol. 6771 of *Lecture Notes in Computer Science*, pp. 391–398, Springer Berlin Heidelberg, 2011.

[54] L. Hargrove, Y. Losier, B. Lock, K. Englehart, and B. Hudgins, "A real-time pattern recognition based myoelectric control usability study implemented in a virtual environment," in *Engineering in Medicine and Biology Society, 2007. EMBS 2007. 29th Annual International Conference of the IEEE*, pp. 4842–4845, Aug 2007.

[55] Intel, "Intel download center," 2015.

[56] L. Seregni, "ERACLE wearable EMG gesture recognition system," Master's thesis, Politecnico di Milano, 2010.

[57] A. R. a. Marco Barbero, Roberto Merletti, *Atlas of Muscle Innervation Zones: Understanding Surface Electromyography and Its Applications*. Springer-Verlag Mailand, 2012.

[58] LoStrangolatore, "Human anatomy planes-it," 2012.

[59] Y. Pan, W. Y. Lin, Y. H. Wang, and K. T. Lee, "Computing multiscale entropy with orthogonal range search," *Journal of Marine Science and Technology*, vol. 19, no. 1, pp. 107–113, 2011.

[60] Y. H. Pan, Y. H. Wang, S. F. Liang, and K. T. Lee, "Fast computation of sample entropy and approximate entropy in biomedicine," *Comput Methods Programs Biomed*, vol. 104, pp. 382–396, Dec 2011.

[61] A. Uncini, "Sintesi dei circuiti adattabili con approccio statistico e informativo," Dispense, Università "La Sapienza" di Roma, 2000.

[62] T. Kohonen, "Self-organized formation of topologically correct feature maps," *Biological Cybernetics*, vol. 43, no. 1, pp. 59–69, 1982.

[63] M. Y. Kiang, "Extending the kohonen self-organizing map networks for clustering analysis," *Computational Statistics & Data Analysis*, vol. 38, no. 2, pp. 161 – 180, 2001.

[64] Intel, "Intel ark," 2015.

[65] G. Gini, L. Cavazzana, F. Mutti, P. Belluco, and A. Mauri, "New results on classifying EMG signals for interfacing patients and mechanical devices," in *New Trends in Medical and Service Robotics*, Rodić, D. Pisla, H. Bleuler (ed), New Trends in Medical and Service Robotics: Challenges and Solutions, 2014.

[66] D. Rivela, A. Scannella, E. Pavan, C. Frigo, P. Belluco, and G. Gini, "Classification of surface EMG through pattern recognition aimed at controlling a powered shoulder prostheses," *Transactions on Biomedical Engineering*, 2015.

[67] G. Gini, P. Belluco, F. Mutti, D. Rivela, and A. Scannella, "Towards a natural interface for the control of a whole arm prosthesis," in *New Trends in Medical and Service Robotics*, vol. 38, H. Bleuler et Al. (eds): Springer series in Mechan. Machine Science, 2015.