SCUOLA DI INGEGNERIA

Intelligent System Project

Professore
B. Lazzerini

Assistente
E. D'Andrea

Studente
Valerio Tanferna

A.A. 2015/2016

Problema

Il progetto richiede l'analisi di un insieme di dati medici. Il contesto di applicazione è la dermatologia e in particolare, la terapia di compressione per mezzo di bendaggi per il trattamento di ulcere venose alle gambe. Le ulcere venose sono lesioni della pelle delle gambe dovute ad una cattiva circolazione del sangue. Per la riparazione delle ulcere, deve essere potenziata la circolazione del sangue, questo lo si fa attraverso un bendaggio che viene applicato con una certa pressione sulla pelle in cui è presente l'ulcera. La pressione applicata sull'ulcera, quindi l'efficienza della terapia, dipende da diversi fattori:

- il tipo di bendaggio;
- la corretta appicazione del bendaggio;
- l'attività svolta dal paziente.

È stato applicato un bendaggio a compressione sul polpaccio e sono stati inseriti 3 sensori in diverse posizioni per misurare la pressione del bendaggio. I dati a disposizione si riferiscono a 10 volontari. Ogni volontario ha indossato il bendaggio per 12 minuti. Durante questo tempo i volontari hanno eseguito diverse attività o mantenuto diverse posizioni, ognuna delle quali è stata svolta/mantenuta per 3 minuti circa. I sensori misuravano la pressione con un tempo di campionamento di circa 82ms. Le posizioni/attività prese in considerazione sono le seguenti:

- dorsiflessione
- salita delle scale
- camminata
- posizione supina

I dati sono organizzati nel seguente modo: è fornita una cartella per ogni volontario. In ogni cartella ci sono 4 files, ognuno riferito ad un'attività svolta. Ogni file contiene la misurazione dei tre sensori (le prime tre colonne) e il relativo tempo (quarta ed ultima colonna). In figura 1 i valori del sensore 1 dell'attività dorsiflessione per il paziente 1.

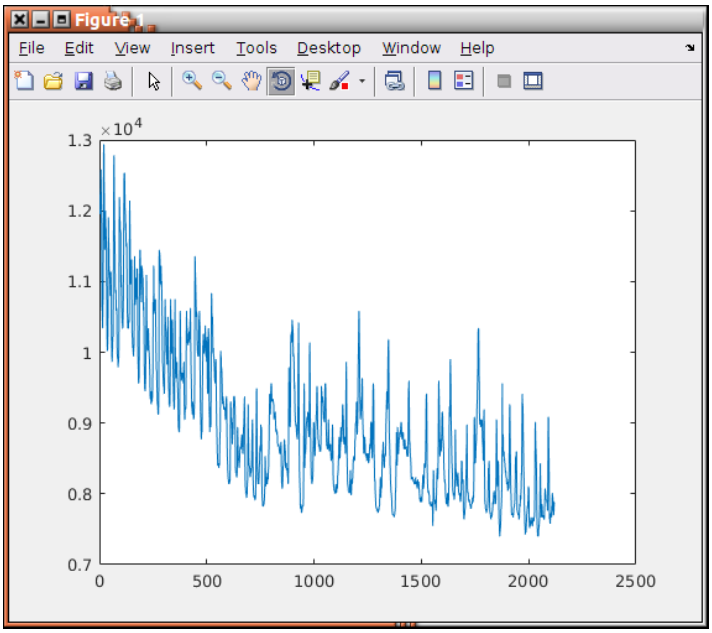

Figura 1

Obiettivo
Individuare la posizione/attività eseguita dal volontario analizzando i valori di pressione del bendaggio per distinguere tra posizione supina, dorsiflessione, camminata e salire le scale. Individuare l'intervallo di tempo più piccolo necessario per identificare la posizione/attività.

Si rendono necessarie delle operazioni preliminari per organizzare i dati in maniera opportuna.
Uno script matlab, main.m si occupa di eseguire tutte le operazioni.

Carica_dati
Questa funzione si occupa di creare una matrice di 10x16 elementi. Si fa uso della xlsread per leggere i file in formato .xslx.
I campioni presenti non hanno tutti la medesima lunghezza, pertanto vengono letti i file dalla riga 1 fino alla 2200. È stato scelto questo valore perchè tutti i dati avevano una lunghezza inferiore a questo valore.
I dati dei 3 sensori, per ognuna delle 4 attività per tutti i volontari (10) vengono caricati in una matrice DATA di dimensioni 10x16, dove 10 sono i volontari e 16 sono i dati a disposizione per ogni volontario (3 sensori + time, per le 4 attività = 16).

Una posizione/attività è identificata dai valori dei 3 sensori, campionati ogni 80ms circa, per 180 secondi. Il numero di features totali è circa 6500 per ogni posizione/attività, cioè i tre sensori diviso il numero di campioni, cioè il tempo totale di campionamento, 180 secondi diviso il tempo di campionamento, 82 millisecondi (3x(180/0,0082)).
I campioni a disposizione sono 40, ovvero le 4 posizioni/attività per i 10 pazienti.
Il numero dei campioni è ancora troppo basso, quindi sarà necessario aumentarli.

Per fare ciò si riduce l'intervallo temporale per l'identificazione di una posizione/attività. L'intervallo originario per ogni attività era di 180 secondi, sono state eseguite varie prove di riconoscimento con diversi intervalli di identificazione. Dopo vari test con intervalli differenti è stato identificato un intervallo di identificazione pari a 30 secondi per ogni attività, il quale è stato ottenuto semplicemente dividendo tale intervallo in 6 gruppi da 30 secondi. I nuovi sottointervalli ottenuti saranno i nostri campioni.

max_min_med
Tale funzione permette un aumento dei campioni ed il prelievo delle informazioni relative ad ogni sotto intervallo, cioè i valori massimo, minimo e la media.
Con la riduzione dell'intervallo di identificazione il numero di sample è incrementato da 40 a 40x6=240.

Purtroppo il numero di features è ancora troppo alto per procedere ad un'analisi.

Come precedentemente fatto, si procede ad una suddivisione di tali sottointervalli. Essi hanno una durata di 30 secondi, che è stata divisa in 5 sottointervalli di esattamente 6 secondi. Da tali sottointervalli di 6 secondi si estrapolano 3 caratteristiche: valore massimo, minimo e medio. In questo modo le features sono passate da circa 1000, cioè da tre sensori per la durata degli intervalli, 30 secondi, diviso il tempo di campionamento di 82 milli secondi, (3x(30/0,0082)) a sole 45 features (3x5x3).

Tale funzione prende come paramentro la matrice contenente tutti i dati. Crea 2 matrici, una per i dati di input di dimensione 240x45 ed una matrice per i target di dimensione 240x4.
Essa scorre tutti i dati, per volontari, posizioni e per ogni sensore scorre gli intervalli di 6 secondi.

Per fare ciò fa uso di 2 indici, affinchè il calcolo dei valori massimo, minimo e medio di un sotto intervallo venga effettuato tramite min(), mean() e max() messe a disposizione da Matlab. Come input gli viene passato un vettore.

La matrice inputs avrà questa composizione:
min|mean|max, ripetuta per 3 sensori e 5 sottointervalli (3x5x3) per le colonne.
Il numero di righe è pari a quella dei 10 volontari per 4 posizioni per 6 intervalli.

Per comodità nelle fasi successive le matrici di inputs e targets sono state suddivise in 2 sottomatrici per il training ed il test.
La suddivisione è stata del 15% e 85% rispettivamente per testing e training.

```
function [dati, target] = max_min_med(DATA)
   intervalli = 6;
   sottointervalli = 5;
        dati = zeros(40*intervalli, sottointervalli*3*3);
        target = zeros(40*intervalli, 4);
        for i=0:intervalli-1
                for v=0:9
                        for p=0:3
                                target(40*i+v*4+p+1,p+1) = 1;
                                for s=0:2
                                        iniziosottointervallo = fix( numel(DATA{v+1,
4*p+s+1})/intervalli)*i+1;
                                finesottointervallo = fix( numel(DATA{v+1,
4*p+s+1})/intervalli/sottointervalli);
                                        for si=0:sottointervalli-1

                                                dati(40*i+v*4+p+1, sottointervalli*3*s+3*si+1) =
min(DATA{v+1,4*p+s+1}(iniziosottointervallo:iniziosottointervallo+finesottointervallo-1));

                                                dati(40*i+v*4+p+1, sottointervalli*3*s+3*si+2) =
mean(DATA{v+1,4*p+s+1}(iniziosottointervallo:iniziosottointervallo+finesottointervallo-1));

                                                dati(40*i+v*4+p+1, sottointervalli*3*s+3*si+3) =
max(DATA{v+1,4*p+s+1}(iniziosottointervallo:iniziosottointervallo+finesottointervallo-1));

                                                iniziosottointervallo =
iniziosottointervallo+finesottointervallo;
                                        end
                                end
                        end
                end
        end
end
```

Adesso i dati sono pronti per la features selection.

sequentialfs
La features selection viene eseguita dalla funzione sequentialfs().
Essa ritorna le features selezionate ad ogni step, sotto forma di vettori binari (matrice), un vettore con le percentuali di errore di riconoscimento.
Fs è un vettore binario con le features selezionate ed una struttura history che contiene una matrice In con i vettori binari delle features selezionate ad ogni iterazione, ed un vettore Crit con le percentuali di errore sul riconoscimento.
Viene definita una fun, la quale è una funzione che implementa tale rete neurale, esegue pattern recognition, che ritorna l'errore di riconoscimento.

In seguito alla features selection, inserisco le features selezionate in altre matrici, input_train_fs e target_train_fs, che sono le matrici per il training della rete neurale.

```
fun = @(trainingX, trainingT, testingX, testingT)compute_errors(trainingX', trainingT', testingX', testingT');
        [fs, history] = sequentialfs(fun, input_train, target_train, 'options', opts);
```

Neural Network
Ottenute le features per la rappresentazione delle attività/posizioni è possibile generare la rete neurale e allenarla. In figura 2 il tools di Matlab per le reti neurali durante il training.

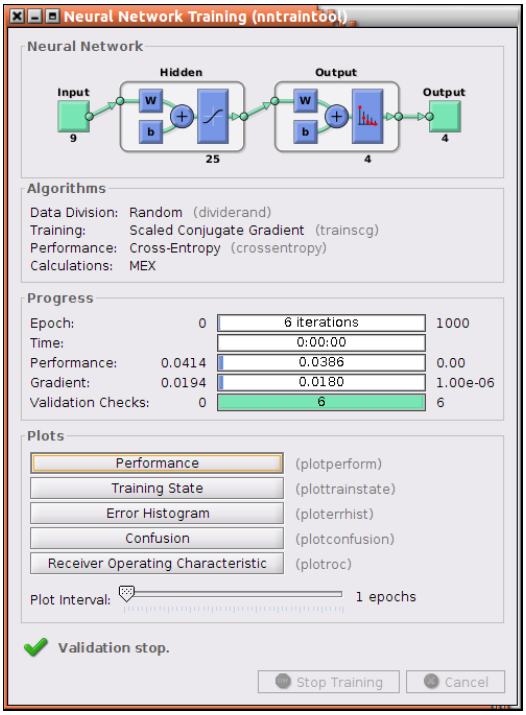

Figura 2

La rete ha 25 neuroni nel livello nascosto ed e' addestrata 10 volte. Ritorna la rete con il miglior riconoscimento, cioè quella con la percentuale di errore minore.
La dimensione del testing set è il 15% del training set a disposizione, cioè 36 valori su 240.

Tale rete neurale ha un tasso di riconoscimento del 90% circa. Tale valore varia di circa il 10% a seconda del testing set scelto nella fase di feature selection.

Esecuzione 1: 85.7%
Esecuzione 2: 92.4%
Esecuzione 3: 97.8%
Esecuzione 4: 88.9%
Esecuzione 5: 96.2%
Esecuzione 6: 93.8%

Nella figura 3 la matrice di confusione.

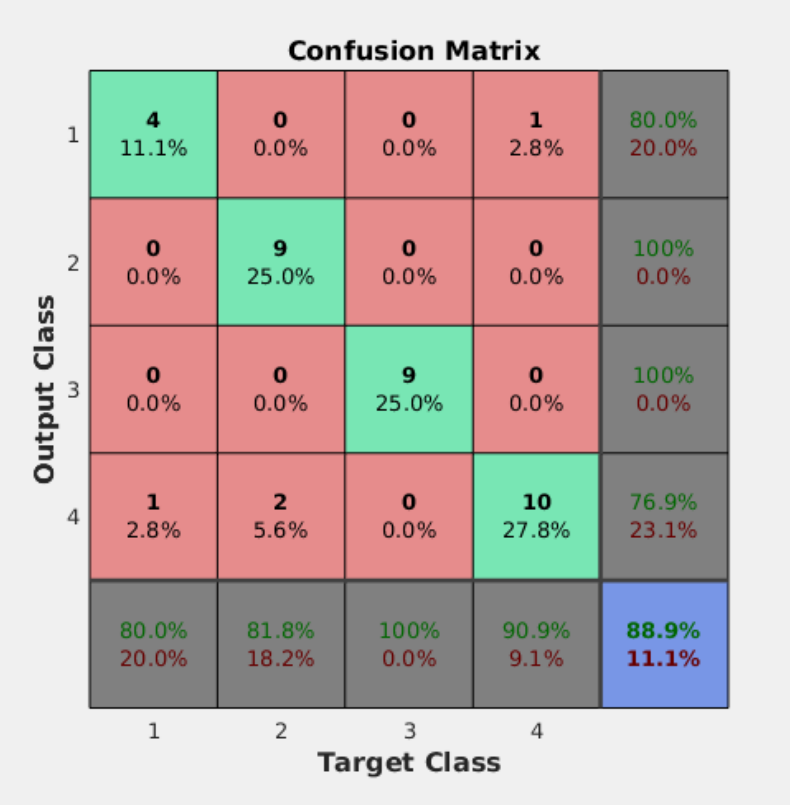

Figura 3

Features
Visualizzazione grafica delle features selezionate nella figura 4.

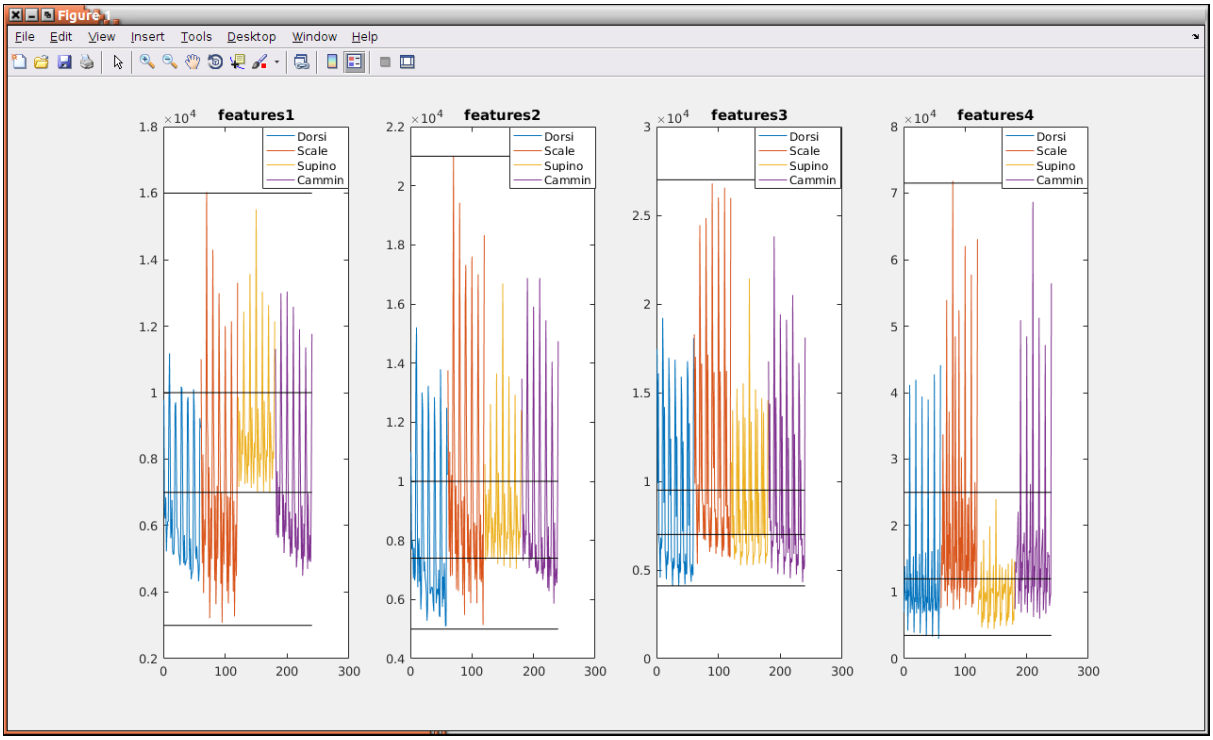

Figura 4

Analizzo le features con una funzione. Essa prende in ingresso 4 features, le sottomatrici per il train delle matrici originarie di ingresso e target, la struttura history ritornata dalla sequentialfs. Dalle matrici di ingresso vengono prelevate solo le features selezionate dalla features selection e se ne visualizza la distribuzione, affinché si possa eseguire un'analisi grafica per definire gli intervalli di definizione delle membership functions del sistema fuzzy.

Attraverso tale funzione è stato possibile vedere la distribuzione dei valori per definire l'intervallo delle membership function e la relativa forma. Per la scelta degli intervalli e la forma è stato utilizzato rispettivamente un approcio visivo e sperimentale.
History è la struttura contenente gli indici delle feature selezionate dallla sequentialfs. Per ridurre il numero delle feature si selezionano solo le prime 4 attraverso gli indici del campo "In" della struttura.

L'intervallo di definizione delle funzioni è:
Features 1: 3000, 7000, 10000, 16000;
Features 2: 5000, 7900, 10000, 21000;
Features 3: 4100, 7000, 15000, 27000;
Features 4: 3500, 12000, 25000, 71500.

Mamdani

Per lo sviluppo del sistema Fuzzy Mamdani è stato utilizzato il tool fornito (avviabile digitando "fuzzy"). Per costruire correttamente il sistema è necessario analizzare le features selezionate dalla features selection, in particolare la loro distribuzione per poter definire gli intervalli di definizione delle regole.
In figura la struttura del sistema fuzzy mamdani.
Come risultato viene raggiunta un accuratezza del 45,25% (train set più test set), valutata attraverso la evalfis() con parametri una matrice di 240x4. In figura 5 il sistema Mamdani.

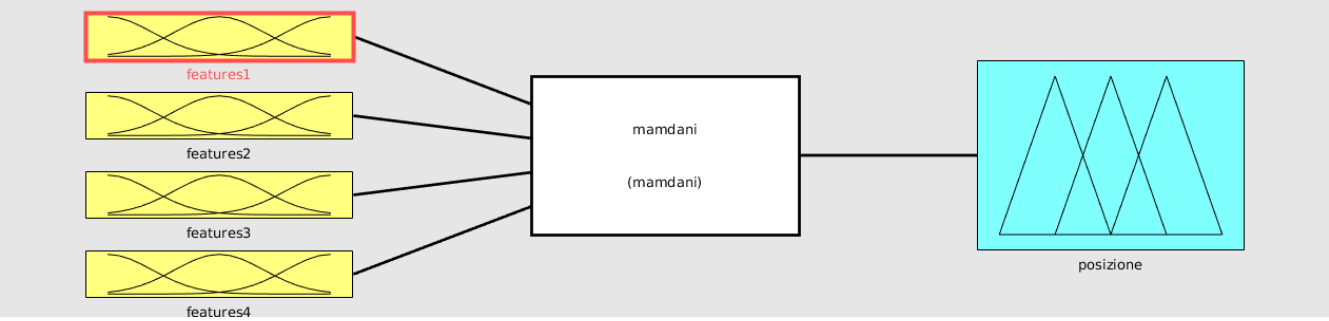

Figura 5

Per far ciò eseguo il plot delle 4 features che verranno utilizzate, tra quelle selezionate dalla sequentialfs.

Per le membership function è stata scelta una forma Gaussiana, di tipo *gauss2mf*. La scelta della forma è stata eseguita in maniera sperimentale provando le varie tipologie, questa forma dava il rilevamento piu' alto.
Da notare le che funzioni membro sono traslate a destra, scartando i primi valori che facevano scendere di molto la percentuale di rilevamento. Nelle immagini la forma delle membership function per le varie features. Nelle figure 6, 7, 8 e 9 sono mostrate le distribuzioni delle membership function per ogni featureas selezionata.

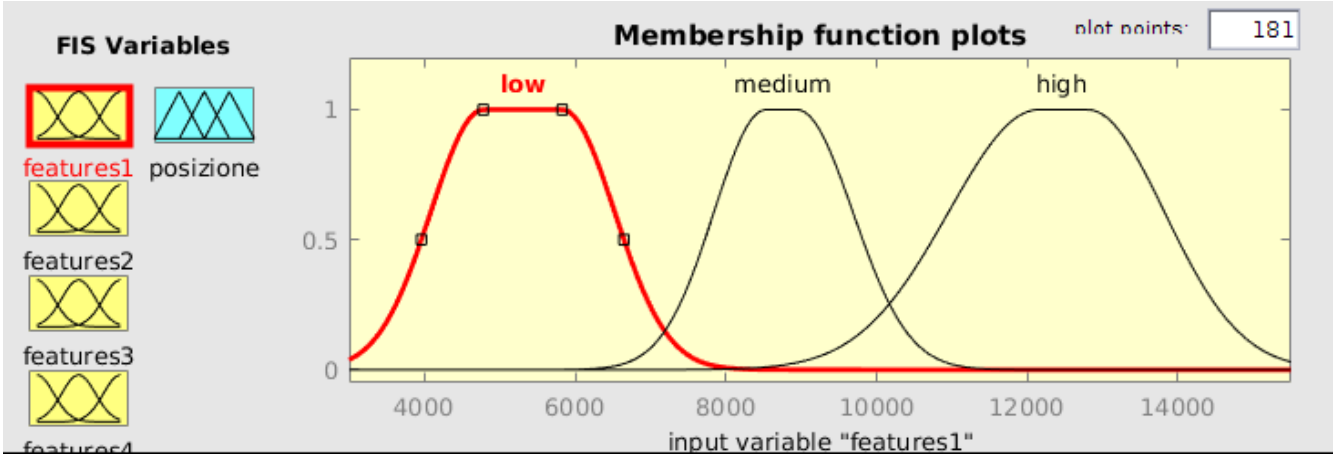

Figura 6

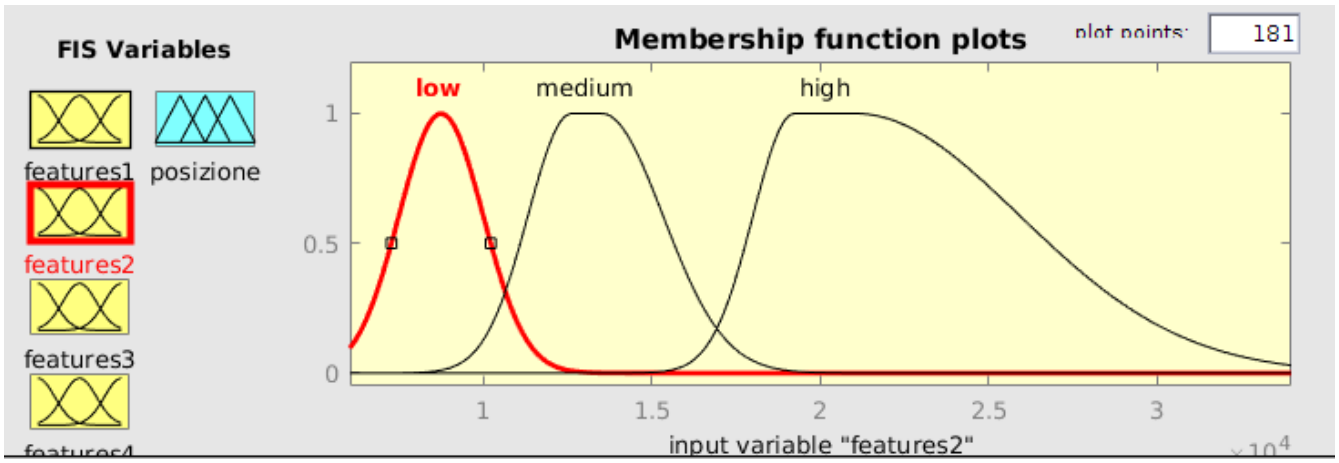

Figura 7

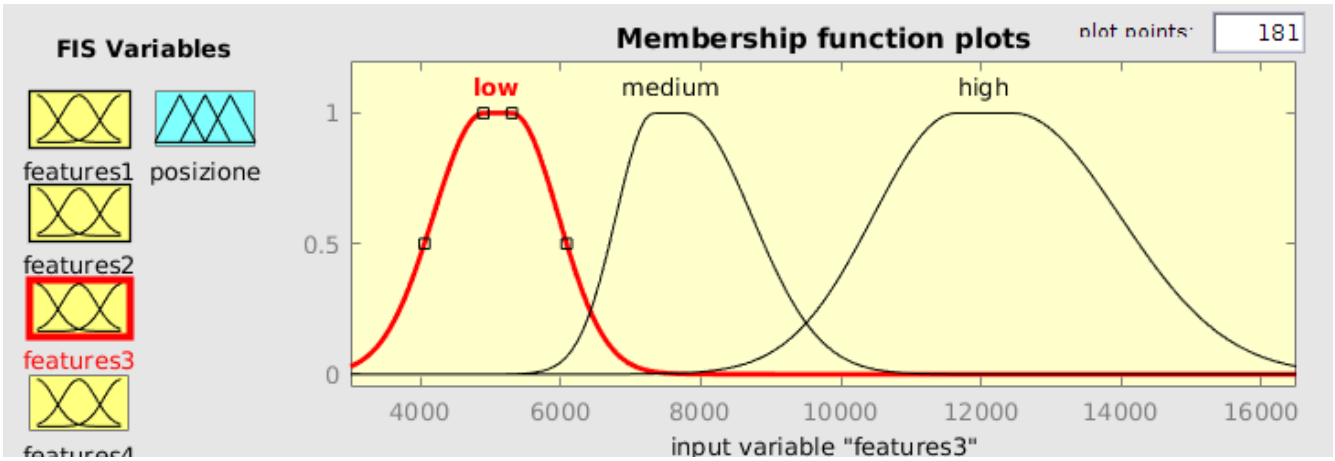

Figura 8

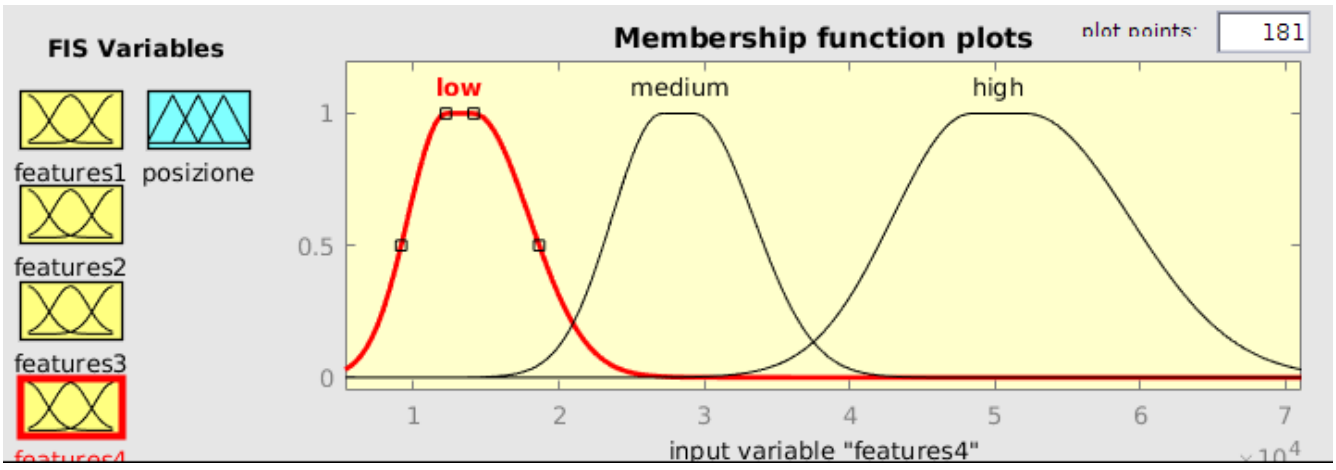

Figura 9

In figura 10 I valori fuzzy per l'uscita.

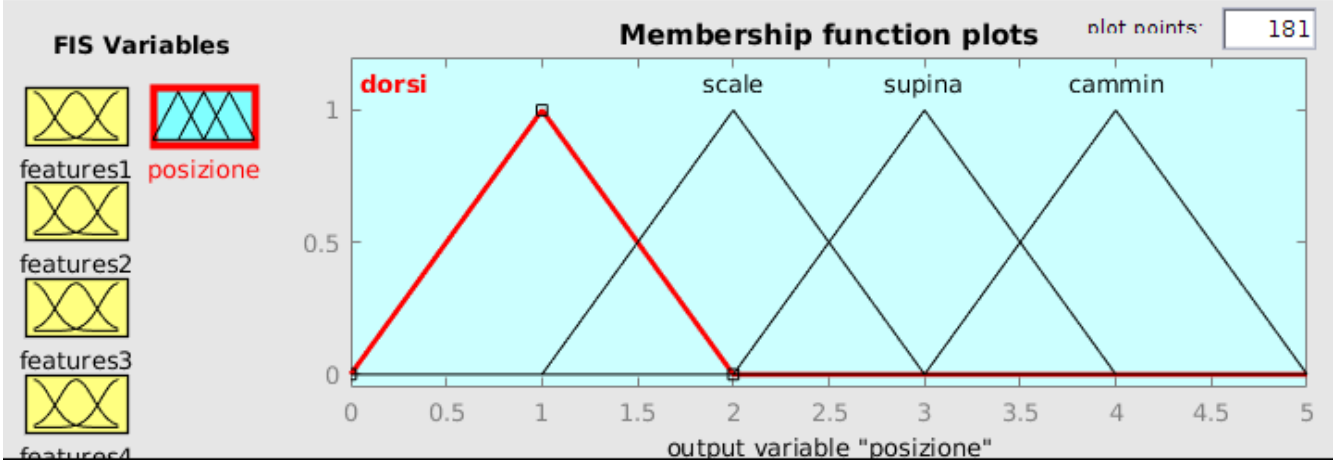

Figura 10

Le regole scelte sono le seguenti:

1. If (features1 is low) and (features2 is low) and (features3 is low) and (features4 is low) then (posizione is dorsi) (0.4)
2. If (features1 is low) and (features2 is medium) and (features3 is low) and (features4 is medium) then (posizione is scale) (0.4)
3. If (features1 is medium) and (features2 is low) and (features3 is medium) and (features4 is low) then (posizione is supina) (0.4)
4. If (features1 is low) and (features2 is low) and (features3 is medium) and (features4 is medium) then (posizione is cammin) (1)
5. If (features1 is medium) and (features2 is medium) and (features3 is medium) and (features4 is low) then (posizione is dorsi) (1)
6. If (features1 is medium) and (features2 is medium) and (features3 is medium) and (features4 is medium) then (posizione is scale) (1)
7. If (features1 is low) and (features2 is low) and (features3 is medium) and (features4 is low) then (posizione is cammin) (1)

Defuzzificazione

Per il sistema fuzzy è stato utilizzato un sistema *Mean Of Maxima*, il quale ci permette di discriminare le varie attività, in quanto ogni qual volta un valore di una funzione è maggiore di un altra il valore salta e discrimina in maniera netta dal precedente. Un metodo Centre Of Maxima potrebbe portare in confusione, dato che il risultato verrebbe discriminato da una soglia troppo bassa di valori e l'uscita del sistema sarebbe stata ambigua con un valore numerico centrato tra le funzioni di uscita generando un risultato errato.

L'operatore di aggregazione scelto e' "max", l'implicazione "prodotto".

Sugeno

Per lo sviluppo del sistema Sugeno-type fuzzy si è utilizzato il tool integrato nell'ambiente Matlab, ANFIS-GUI-tool. Per invocare il tool è sufficiente digitare "*anfisedit*".
È necessario predisporre i dati in maniera opportuna: i dati di ingresso per il train del sistema sono quelli selezionati dalla features selection, quelli per il target sono valori che vanno da 1 a 4 per le posizioni. Questi dati sono stati concatenati in un'unica matrice: nelle prime 4 colonne sono presenti le 4 features utilizzate, nell'ultima un vettore per identificare la posizione.

Per generare il sistema si sono utilizzate le seguenti configurazioni: Grid partition che genera una FIS a singola uscita Sugeno-type.

Le configurazioni scelte sono le seguenti:
Per il train FIS, il metodo scelto per la rete per l'apprendimento è un sistema ibrido, che fa uso del metodo dei minimi quadrati e del backpropagation gradient descent, il quale garantisce un errore più basso rispetto al training con solo il backpropagation.
Criteri di stop per la fase di training:
Il numero di epoche scelto è 10. L'errore all'aumentare del numero di epoche diminuisce di valori molto piccoli (dell'ordine del 0.0002 circa per ogni epoca).
La soglia di errore dell'allenamento è 0 in quanto l'obbiettivo è quello di minimizzare l'errore.

Per l'input il numero di membership function è pari a 7 di tipo *gaussmf*, cioè che definisce una forma Gaussiana, il tipo di funzione per l'uscita è costante.

La struttura Anfis risultante è mostrata in figura 11:

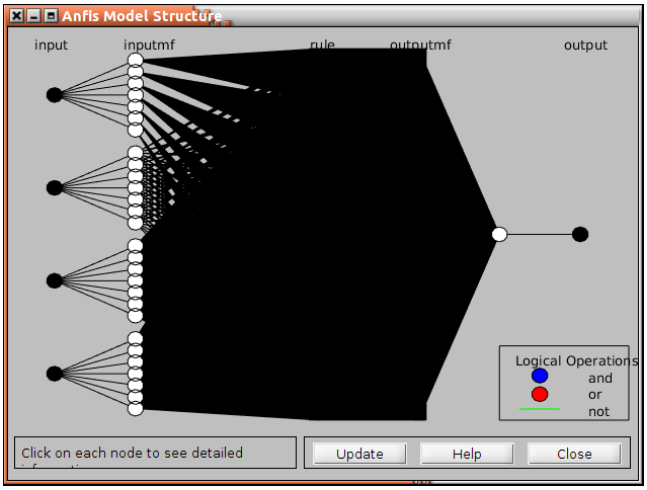

Figura 11

Sono stati effettuati vari test con varie configurazioni, quella scelta offre un buon trade-off tra prestazioni e risultati.

Con un numero di Mfs pari a 7 per ognuno dei 4 input si ottiene un valore dell'errore pari a 0,35844. In figura 12 l'andamento dell'errore nelle varie epoche durante l'addestramento.

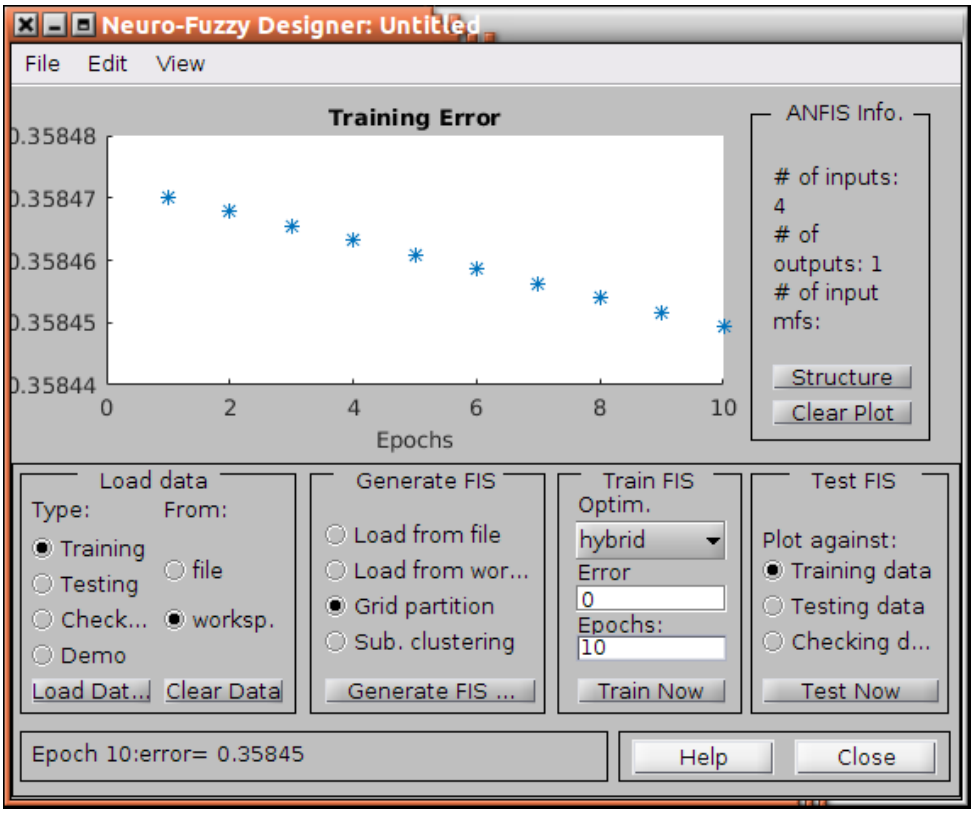

Figura 12

Nel main sono presenti le seguenti righe di codice:

```
fuzzy_sugeno = genfis1(sugeno_train, 7, 'gaussmf', 'constant');
fuzzy_sugeno = anfis(sugeno_train, fuzzy_sugeno);

sugeno_test_recognition = sugeno_recognition(input_test, target_test, feature_sel, fuzzy_sugeno);
sugeno_train_recognition = sugeno_recognition(input_train, target_train, feature_sel, fuzzy_sugeno);
```

dal quale, la percentuale del sistema Sugeno è:
90.5% per il train set e del 39% per il test set.
In figura 13 l'uscita del sistema rispetto ai dati di training.

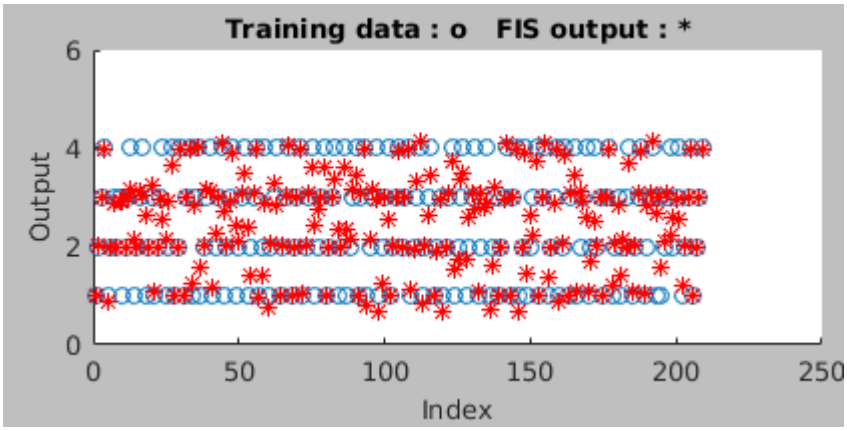

Figura 13

Conclusioni

Lo scopo del progetto era creare tre sistemi differenti che permettessero di identificare e classificare le posizioni date di alcuni volontari, usando le infomazioni derivate dai dati iniziali.
I tre sistemi creati sono una rete neurale, un sistema di inferenza Mamdani ed un sistema Sugeno, i quali analizzano i dati nel dominio del tempo.
L'intervallo minimo di determinazione di un'attività è di 30 secondi, che permette di identificare una features. Il sistema Sugeno raggiunge dei livelli di rilevamento buoni per un'applicazione reale.