UNIVERSITÀ DEGLI STUDI DI PISA

Facoltà di Matematica Fisica e Scienze Naturali
Laurea Specialistica in Tecnologie Informatiche


# Sviluppo di un sistema di controllo per sensore tattile skin-like

Candidato
Domenico D'Antonio

Relatore
Antonio Cisternino

A.A. 2006-2007

# Indice

# Introduzione

Tra i cinque sensi il tatto è uno dei più estesi. Attraverso esso si familiarizza (conosce) con l'ambiente circostante, si instaura una sorta di rapporto con tutto ciò che ci è vicino, che ci è a "portata di mano", attraverso esso riusciamo ad acquisire molte informazioni sul mondo esterno che ci circonda ed è un valido ausilio per tutti gli altri sensi. Nel campo della ricerca per facilitare l'usabilità del software, l'aspetto del contatto diretto con la macchina è molto importate, questo perché il contatto fisico rende più reale e vera l'interazione, donandole familiarità ed a volte velocizzando alcune delle operazioni che l'utente compie, ad esempio associando a gesti spontanei delle operazioni da eseguire.

Lo scopo di questa tesi è quello di realizzare un software in grado di classificare alcune delle sensazioni tattili riconosciute dalla pelle, ovvero la pressione e lo sfregamento da parte di un corpo sul sensore tattile skin-like, al fine sviluppare un software in grado di riconoscere questo tipo di interazione.

Il presente lavoro è articolato in diverse fasi di sviluppo: la prima è un'analisi di tutto il sistema per capire quali sono i moduli principali che lo compongono, come metterli in comunicazione, il tipo di dati che vengono scambiati e come far riconoscere questi dati al software per la classificazione; nella seconda fase si è passato ad analizzare ognuno dei blocchi che compongono il sistema, raffinando sempre più lo studio sino ad arrivare ad un'implementazione funzionante di ognuno di essi con lo scopo finale di riconoscere la pressione e lo strofinamento; l'ultima fase consiste nel testare il software per capire come classifica i segnali ricevuti, qual è il margine d'errore commesso e come diminuirlo per far aumentare la precisione nel classificare i dati. Alla fine di quest'analisi si può affermare che il sistema può essere raggruppato in tre moduli principali: il primo è il sensore skin-like con la sua scheda di interfacciamento; il secondo modulo è la scheda WildFire 5282 che permette di campionare i dati ricevuti dalla pelle ed inviarli attraverso una connessione LAN al pc (in realtà questa scheda può far cose ben più complesse che verranno analizzate in seguito). L'ultimo modulo consiste nel software sviluppato per ricevere, elaborare e classificare i dati inviati dalla scheda WildFire, esso è composto da diversi tool, uno dei quali è il classificatore SVM (Support Vector Machine [SVM1]) che permetterà di riconoscere le diverse tipologie di interazioni con il sensore.

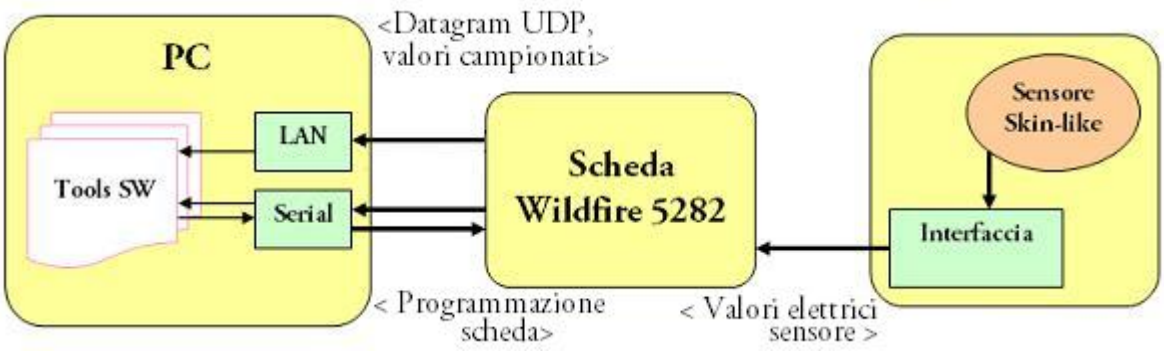
Il presente elaborato è stato inoltre diviso in sette sezioni: nel primo capitolo si descrive la struttura generale del sistema al fine di dare in idea di tutta la struttura e delle connessioni tra i diversi moduli, subito dopo vengono descritte le caratteristiche principali del sensore skin-like, cosa è in grado di rilevare, la sua sensibilità ed i limiti di utilizzo. In un secondo

momento viene descritta la scheda WildFire 5282, le principali funzionalità offerte e com'è possibile interagire con essa. Successivamente viene presentata un'introduzione sulle Support Vector Machines e sulle librerie disponibili su internet, viene illustrato tutto il tool di controllo creato per implementare il classificatore ed infine vengono riportati i test eseguiti sul classificatore, cercando di coprire il maggior numero dei casi di utilizzo del sensore, allo scopo finale di valicare il classificatore. In ultimo vengono tracciate le conclusioni sul lavoro svolto e gli sviluppi futuri.

# Capitolo 1

# La struttura del sistema

Il sistema ad una prima approssimazione, può essere strutturato in tre moduli principali: il primo modulo corrisponde al sensore, il secondo alla scheda wildfire 5282 e l'ultimo rappresenta tutti gli applicativi sviluppati nel corso della tesi.




(Figura 1: La struttura del sistema)

In questa parte vengono descritte in linee generali le principali funzionalità di ogni modulo e l'utilizzo fatto all'interno del lavoro.

Il **modulo sensore** è composto a sua volta da due componenti: il sensore skin-like vero e proprio ed una scheda di interfacciamento con esso. *É* bene puntalizzare che il sensore skin-like è composto a sua volta da una griglia di resistenze, con valore variabile da 1,1MΩ a 1,9 MΩ (in base alla pressione effettuata sul sensore ed alla temperatura dello stesso), mentre la scheda wildfire 5282 ha come input una tensione nel range 0-5V. L'interfaccia al sensore serve proprio per alimentare le resistenze e fornire in uscita una

tensione compatibile con l'input della scheda Wildfire. Quindi con resistenze del ordine di $10^6$, secondo la legge di Ohm: $V = R \times I$, occorre alimentare le resistenze con valori dell'ordine di $10^{-6}$. Produrre questo genere di correnti con componenti standard, vuol dire essere soggetti a disturbi del segnale. Questo fenomeno è particolarmente evidente, quando si effettua sul sensore una pressione maggiore del semplice sfioramento, la resistenza interessata dalla pressione subisce un cambiamento evidente, mentre le altre resistenze anche pur subendo minime variazioni (dovute alla vicinanza col punto di contatto) hanno comunque una variazione del segnale evidente e ben maggiore rispetto a quella attesa. Quando la pressione esercitata sul sensore è considerevole, questa variazione è presente anche nelle resistenze che sono distanti dal punto di pressione, le quali effettivamente non modificano il proprio valore resistivo, ma il segnale inviato dall'interfaccia subisce comunque un cambiamento. Questo perché lavorando con correnti molto basse, variazioni di tensione degli ingressi si riperquotono su tutto il circuito (in generale questo tipo di disturbi è sempre presente in circuiti del genere, però è molto piccolo tant'e che lavorando nell'ordine dei $10^{-2}$ o $10^{-3}$ questo fenomeno non è visibile). Questo effetto, viene poi diffuso su tutti gli integrati presenti nell'interfaccia, perché essi condividono la stessa alimentazione, anche se in forma minore. Per quanto riguarda l'alimentazione della scheda, essa è inoltre soggetta al disturbo dei 50Hz della rete elettrica, però questo fenomeno viene eliminato attraverso il software, impostando la frequenza di campionamento proprio su 50Hz.

Il **modulo Wildfire 5282**, viene utilizzato come campionatore del segnale proveniente dal sensore tattile, con una frequenza di 5ms, sarà poi il software ad affettuare un sub-sampling ulteriore del segnale, fino ad ottenere una frequenza di campionamento di 20ms (cioè 50Hz). In questo modo è possibile diminuire il tempo di risposta, a discapito di un segnale meno pulito. La scheda è stata programmata utilizzando il compilatore rilasciato dalla casa produttrice, che utilizza il linguaggio C. La scheda svolge la funzionalità di campionare il segnale dagli otto canali d'ingresso, e d'inviare i valori letti su rete LAN utilizzando il protocollo UDP. Questi valori rappresentano le tensioni d'ingresso (al convertitore), nella scala dei millivolt, per questo il range parte da 0 fino a 5120 con una precisione di 5, questo perché la scheda Wildfire ha una precisione di 10 bit. La scelta di utilizzare la scala dei millivolt, permette di effettuare un confronto rapido tra i valori ricevuti dal software, e quelli letti con un tester direttamente sull'interfaccia del sensore.

L'ultimo modulo rappresenta il cuore del lavoro svolto, ovvero i **Tools sviluppati** per poter: elaborare i segnali ricevuti dal sensore, visualizzarli, salvarli e poter istruire il classificatore SVM. I tool sviluppati sono tre: il primo permette di fare preprocessing dei dati ricevuti, in modo tale da ridurre i

disturbi introdotti dall'hardware (delle schede) e per semplificare la vita del classificatore. L'interfaccia grafica è composta da diverse funzioni componibili tra loro, attraverso le quali è possibile amplificare i valori, impostare dei filtri o degli offset, effettuare la derivata (in questo caso il rapporto incrementale), visualizzarli in realtime, salvarli sul disco, ed effettuare predizioni realtime selezionando un classificatore. Grazie alla totale componibilità, si ha la possibilità di raffinare con molta cura l'input, al fine di ottenere dati molto puliti e regolare la sensibilità del classificatore. Il secondo tool, è un editor che prende in input i dati salvati su file del precedente tool, li visualizza in un grafico e premette di etichettarli e salvarli. Per poi essere utilizzati sia per l'addestramento del classificatore SVM, sia come prima verifica nella classificazione. È importante notare come ad ogni etichetta corrisponda una classe che l'SVM dovrà riconoscere, ed il numero di etichette è configurabile dall'utente. L'ultimo tool prende in input il file modificato dall'editor, dopo di che crea un file con il quale si effettuerà il training (train.txt impostazione di default) ed un file per verificate il grado di successo del SVM (predict.txt di default). È possibile configurare entrambe le operazioni, scegliendo di prendere dati casuali dal file oppure in sequenza, impostando un bilanciamento dei dati in base alla classe di appartenenza (ad esempio aumentando nel file di training i dati relativi ad una classe, si aumenta la sensibilità nel identificare dati come appartenenti a quella classe, anche nei confronti di dati che non lo sono, ma sono ma sono vicini come valore). Alla fine di questo processo di creazione di file è possibile effettuare il trainig vero e proprio su diverse configurazioni di SVM e verifica di ognuna di esse, tutto in modo automatico. Inoltre viene generato un report per facilitare la lettura dei risultati prodotti, e si offre la possibilità visualizzare i casi di errore in un grafico, in questo modo è possibile individuare una probabile causa dell'errore, ad esempio associandolo a particolari valori di ingresso, e quindi provare a rimediare ad esso. Per facilitare l'utilizzo dei tool è stato previsto un file di configurazione in XML, nel quale l'utente può impostare sia le diverse tipologie di SVM da utilizzare, sia i dati relativi a file e directory, sia le configurazioni interne dei tool. Per quanto riguarda la classificazione realtime, è stata prevista la possibilità di scegliere tra i diversi SVM presenti nel file XML.

In seguito saranno analizzati più in profondità tutti questi aspetti, sia l'hardware sia il software, descrivendo le caratteristiche di ognuno di essi e le funzionalità offerte ai fini di questo lavoro.

# Capitolo 2

# Il sensore tattile

Il sensore tattile utilizzato, è stato sviluppato in un progetto di tesi nel campo biomedico. L'obiettivo principale del sensore tattile Skin-like è quello di rilevare un semplice sfioramento della sua superficie, una forza applicata e il punto di applicazione. Per la realizzazione del prototipo sono stati seguiti due criteri fondamentali: *la biocompatibilità*, quindi la possibilità di utilizzarlo nel campo biomedico come: robot con percezione tattile, per riabilitazione, per protesi o interventi a distanza in ambito medico. Il secondo criterio riguarda le *proprietà meccaniche*, ovvero una buona robustezza fisica, all'usura (derivata dall'uso e dallo stress), e proprietà elastiche adeguate all'utilizzo. Un aspetto molto interessante che è bene sottolineare, è la differenza tra sensore e trasduttore, un sensore ha come output la variazione della grandezza fisica (per la quale è stato progettato), mentre un trasduttore riporta la misura della grandezza fisica da misurare. Questo aspetto è molto importante, garantisce che la funzionalità del sensore non dipenda (non è legata alla) dalla superficie sulla qualche è posto (ad esempio un piano, un oggetto sferico, una superficie ondulata o quant'altro), ma esclusivamente dalla pressione effettuata su di esso.
La fonte principale alla quale fa riferimento questo capitolo è [Eri06], per quanto riguarda le caratteristiche del sensore e le sue proprietà. Mentre per la parte riguardante l'interfaccia di connessione con esso, sono stati eseguiti test di laboratorio per verificare effettivamente il suo funzionamento, e quali rumori nel segnale sono presenti. Dal lavoro di tesi [Eri06] sono stati realizzati tre prototipi, ognuno rispettante le caratteristiche di biocompatibilità, ma con diverse caratteristiche fisiche. Per questo lavoro di tesi è stato scelto uno tra i tre prototipi sviluppati, la scelta è stata fatta analizzando quale tra loro, avesse le migliore caratteristiche sia meccaniche che elettriche, basandosi sui dati sviluppati nel lavoro [Eri06].
Di seguito segue una descrizione del prototipo utilizzato, quali sono le sue

caratteristiche fisiche ed elettriche, i parametri di utilizzo (come ad esempio, la massima pressione rilevabile) ed i limiti strutturali (pressione massima, oltre la quale si danneggia il sensore, ed usura). Questi dati saranno in seguito utilizzati durante la fase di test dei tool prodotti, in modo da avere un indicazione sui quali sono i valori da aspettarsi ancor prima di eseguire il test, per poi confrontarli con i dati ottenuti. Un altro punto importante riguarda la realizzazione fisica del sensore, nella documentazione [Eri06] è stata riportata anche la procedura per la sua realizzazione, per cui è possibile riprodurre il sensore al fine di migliorare ulteriormente le sue caratteristiche o sviluppare sensori su misura per determinati casi d'uso.

## 2.1 Descrizione del sensore e proprietà fisiche

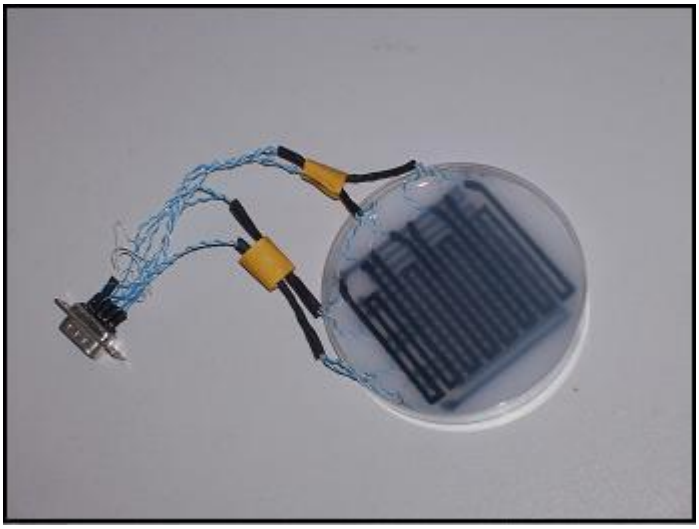
Il materia scelto per realizzare il sensore è il silicone Burman, perché presenta specifiche caratteristiche: meccaniche (flessibile, deformabile, elastico, basso modulo di Young), fisiche (isolante, resistenza alle alte e basse temperature, resistenza all'ossidazione e all'idrolisi), buona biocompatibiltà, lavorabilità e basso costo. Di questo sono stati scelti due diversi tipi in base alla funzionalità di utilizzo:

- un *silicone isolante* per il supporto,
- un *silicone conduttore*, cioè drogato con particelle conduttrici, utilizzato per realizzate le tracce conduttive.

La struttura del sensore è formata da:

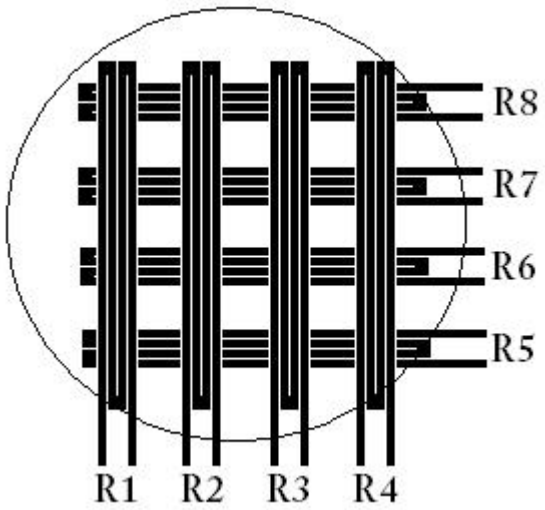
- un primo strato di materiale isolante, il supporto
- su questo vengono fabbricate delle serpentine, in materiale conduttore,
- un secondo strato di materiale isolante,
- una nuova serie di serpentine, questa volte ordinate ortogonalmente alle prime,
- un terzo strato isolante.

La figura che segue è il sensore così ottenuto, ed utilizzato nel lavoro di tesi:



(Figura : Fotografia del prototipo di sensore skin-like utilizzato)

Tale sensore è stato in seguito connesso ad un connettore a 9 poli (visibile nella figura), attraverso il quale verrà in seguito connessa la scheda per l'interfacciamento. La disposizione matriciale delle resistenze all'interno del sensore, permette di individuare il punto di applicazione della pressione sul sensore. Per semplicità, ci riferiremo a tali piste resistive con la seguente nomenclatura:



(Figura : Schema delle tracce conduttive all'interno del sensore)

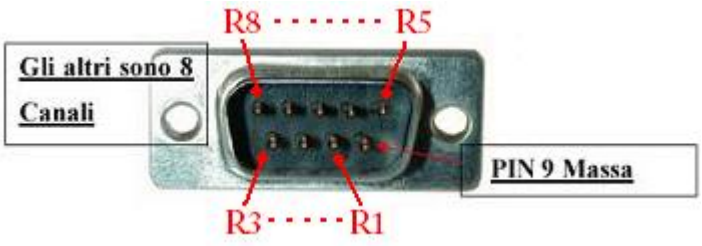
Le resistenze da R1 a R4 sono quelle nel primo strato (più in alto, disposte in verticale), mentre le resistenze da R5 a R8 sono nel secondo strato (più profondo, disposte in orizzontale).
La resistenza R di una pista conduttiva è data dalla seguente formula:

$$R = \frac{(\rho \times \ell)}{S}$$

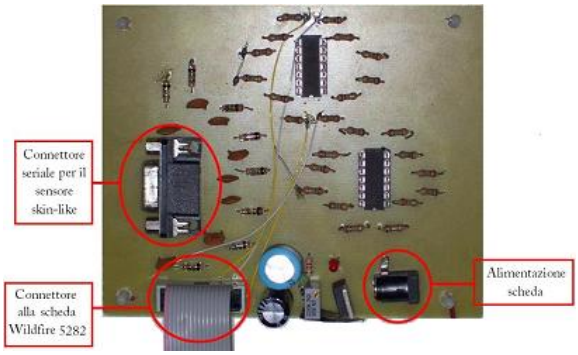
dove $\rho$ è la resistività del materiale di cui è costituito il filamento, $\ell$ è la lunghezza del filamento ed S è la sua sezione. La deformazione dell' oggetto provoca una variazione $\Delta$l della lunghezza del filamento, e quindi una variazione $\Delta$R della resistenza. Nel prototipo utilizzato, si ha una lunghezza delle tracce di 5.5cm ed una larghezza di 2mm.
La lettura dal sensore si traduce nella misurazione delle resistenze per ogni traccia conduttiva al suo interno. Questa misura è effettuata facendo attraversare la resistenza da una corrente nota, dopo di che si misura la caduta di tensione su di essa. Questa misurazione è realizzata dalla scheda di interfacciamento, collegata al sensore attraverso la porta seriale.

## 2.2 Interfacciamento hardware

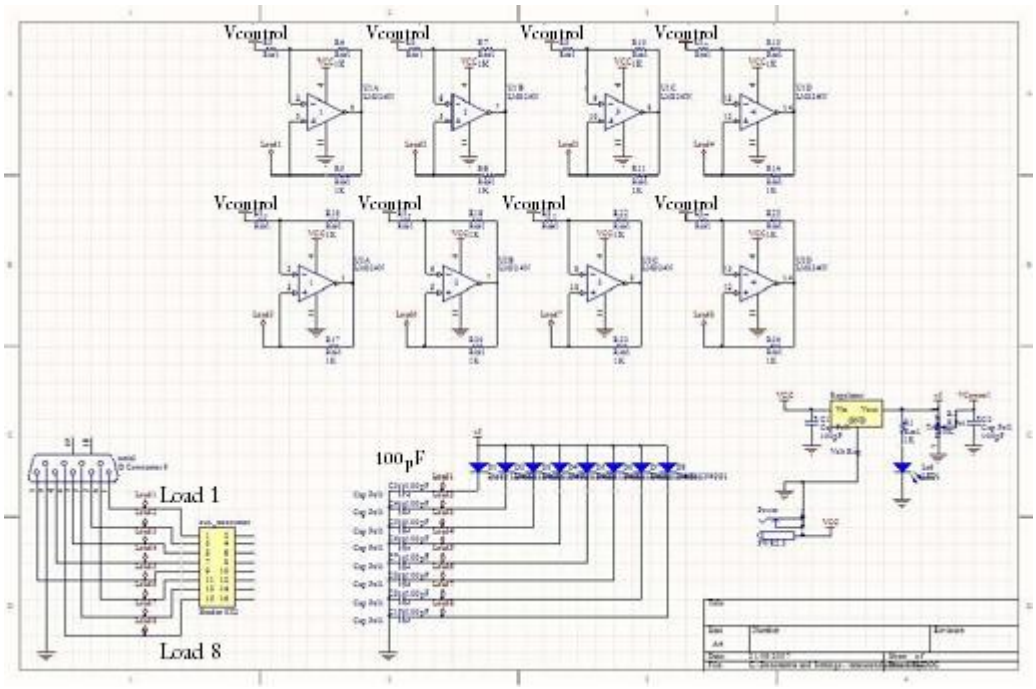
L'interfacciamento al sensore è realizzato attraverso una scheda integrata, la quale è stata realizzata durante il lavoro di sviluppo del sensore, nel dipartimento di Ingegneria Biomedica. Le principali caratteristiche della scheda sono: il *connettore* utilizzato per il collegamento al sensore e la funzionalità propria del *circuito integrato*, di seguito sono riportati i loro schemi elettrici:




(Figura : Schema del connettore a 9 poli collegato al sensore tattile)




(Figura : Fotografia della scheda d'interfacciamento con il sensore. )




(Figura : Schema elettrico del circuito transconduttivo, del collegamento connettore canali, dello stabilizzatore di tensione. )

Tramite un circuito transconduttivo viene generata una corrente costante, attraverso la quale è possibile rilevare il valore della tensione. In questo modo, conoscendo la corrente I e la tensione misurata V, si può ottenere il valore della resistenza R delle singole tracce. Nella figura soprastante viene inoltre mostrato il collegamento tra il connettore a 9 poli e i canali Load1, Load2,..., Load8, inseriti nel cavetto dal numero 1 all'8, mentre i numeri dal9 al 16 sono la massa. La Vcontrol insieme ai Load rappresentano la due Vin dell' amplificatore transconduttivo. La VControl è ottenuta mediate un circuito stabilizzatore di tensione.
Occorre prestare attenzione al contatto tra filo del connettore a 9 poli e la traccia resistiva, a volte la connessione fisica tra essi non è perfettamente stabile dato il basso spessore e l'elevata deformabilità del supporto in silicone, per cui per l'utilizzo di questo prototipo è stata prestata molta attenzione nel verificare che questo contatto non sia danneggiato. Uno dei problemi di questo circuito sono i rumori da cui è affetto, questo si ha perché tutti gli integrati utilizzati per generare la corrente costante risiedono sullo stesso circuito integrato, condividendo la stessa alimentazione. Cerchiamo di capire meglio, visto che la resistenza do ogni pista ha un valori nominali dell'ordine dei $10^6\Omega$, e l'ingresso della porta del convertitore WildFire 5282 lavora in un range di $0 - 5V$, occorre generare una corrente dell'ordine di $10^{-6}$A. Quando si esegue una pressione sul sensore si ottiene come risultato una variazione della resistenza e quindi della tensione, gli operazionali che generano la corrente costante, subiscono l'effetto della variazione di tensione variando di poco il valore della corrente prodotta, questo perché risiedono tutti sullo stesso circuito integrato. Tale disturbo si inizia ad avvertire quando viene a presentarsi una variazione di tensione di circa 0.3V a salire, come effetto finale si ha una variazione minima da parte dei valori letti su tutte le resistenze, interpretata erroneamente come una pressione anche sulle altre piste resistive.
Per eliminare questo disturbo occorre alimentare in modo singolo ogni operazionale, eliminando così il disturbo generato dalla variazione di tensione tra gli operazionali.
Un altro problema da cui è affetta questa scheda è la presenza dei 50Hz dell'alimentazione, il circuito stabilizzatore che alimenta tutti gli integrati, oltre a raddrizzare la corrente e stabilizzarla ha anche un filtro passa basso per eliminare il disturbo dei 50Hz. Purtroppo questo disturbo non è eliminato perfettamente, in questo modo tutti i valori ottenuti sono ne sono affetti. Questo rumore può essere eliminato agevolmente effettuando un ulteriore campionamento da parte del software, sulla frequenza da eliminare. Il segnale così ottenuto risulta stabile e diventa utilizzabile per il nostro scopo.

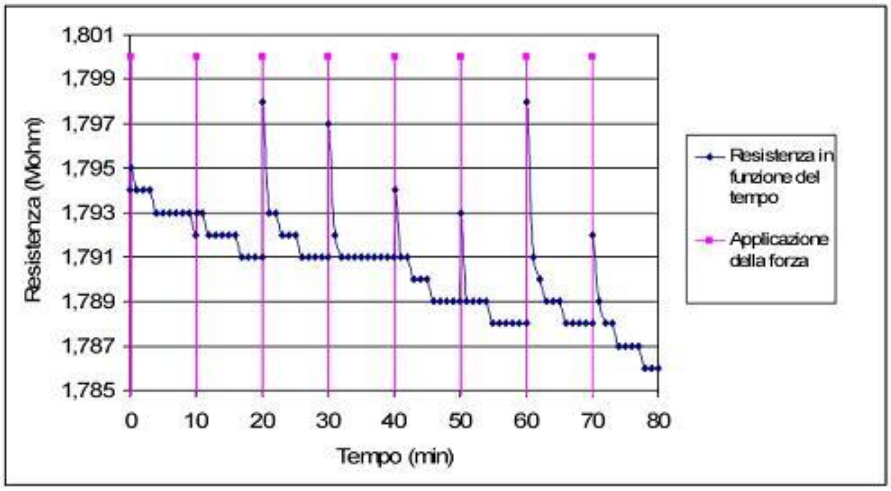
## 2.3 Caratterizzazione del sensore

Di seguito sono riportati i grafici caratterizzanti il sensore, per quanto riguarda l'andamento delle piste resistive nel tempo, nella temperatura ed allo stress.

### 2.3.1 Andamento Resistenza - Peso

I dati relativi a questo test sono riferiti al primo prototipo realizzato, la differenza principale tra esso ed il prototipo utilizzato nella tesi e nello spessore del supporto. In dettaglio, il sensore utilizzato nella tesi ha uno spesso di 1cm, mentre quello utilizzato nel seguenti test ha spessore di 2cm. Ciò comporta, in linea teorica, come conseguenza più importante una diminuzione della sensibilità alla pressione, infatti avendo strato maggiore di supporto, occorre una pressione maggiore per deformare la pista resistiva. Comunque il comportamento generale del silicone risulta essere lo stesso, quindi i risultati riportati di seguiti sono tutti validi anche per il sensore con spessore di 1cm (ma con valori leggermente differenti).

Nei primi esperimenti, rispetto a quelli successivi, manca il rilevamento di una misura corrispondente al momento in cui viene lasciato cadere il peso sul sensore. Il peso infatti, viene rilasciato sul sensore e fatto rimane su di esso per 5 secondi, e poi essere sollevato; solo al quinto secondo si rileva il valore della resistenza del sensore. I pesi utilizzati partono da 1g fino a 700g, sono riportati nel seguente grafico i risultati ottenuti:

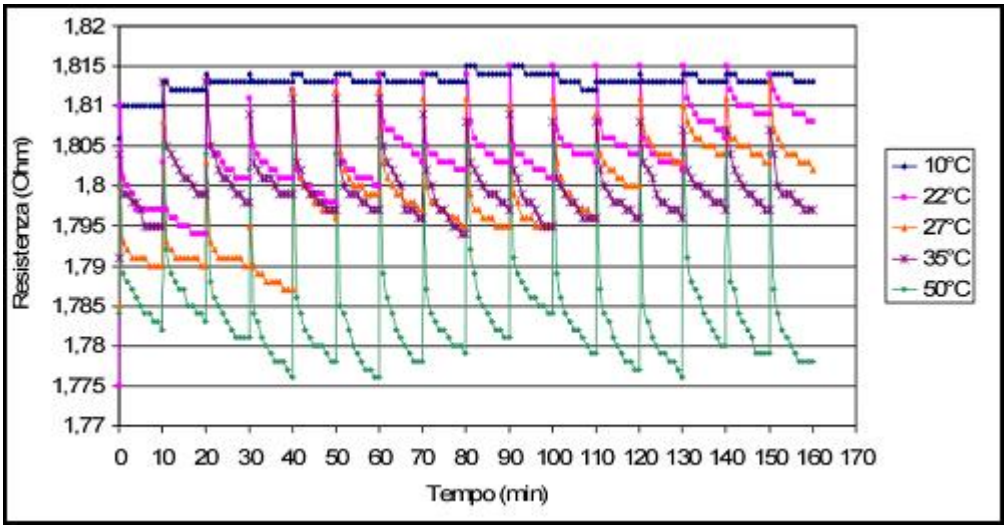



(Figura : Grafico resistenza - tempo )

Dall'osservazione del grafico, si può dedurre che, con l'applicazione dei pesi di pochi grammi, si rilevano variazioni delle resistenze poco evidenti. Sono invece significative le misurazioni per pesi a partire da 100 g, corrispondenti al ventesimo minuto in poi. Infatti, in corrispondenza dell'applicazione del peso, si ha un picco rilevante un minuto dopo, si evidenzia una notevole variazione della resistenza tendente al valore assunto precedente all'impulso.

### 2.3.2 Variazione sensibilità - temperatura

Questo esperimento è stato eseguito sempre sul primo prototipo. Dall'esperienza nell'esperimento precedente, si è visto che per valori piccoli di peso (1 g - 10 g) non si ha una risposta evidente, allora il range di pesi utilizzati parte da 25 g fino a 400 g, con incremento di 25 g per volta. Anche in questo caso, durante l'esperimento, è stata rilevata una resistenza a riposo al quinto secondo, momento in cui si solleva il peso, e ad ogni minuto per i successivi 10 minuti. Dopodiché si aumenta il peso e si ripete l'esperimento. Il dati presenti nel seguente grafico sono stati rilevati da un'unica traccia, R1, in un determinato punto, la Zona centrale, alle temperatura di 10°C, 22°C, 27°C, 30°C, 35°C fino a 50°C, variando il peso applicato. Graficando il variare della Resistenza in funzione del tempo, si può notare come la sensibilità del sensore vari al variare della temperatura, in particolare come diminuisca al diminuire della temperatura.

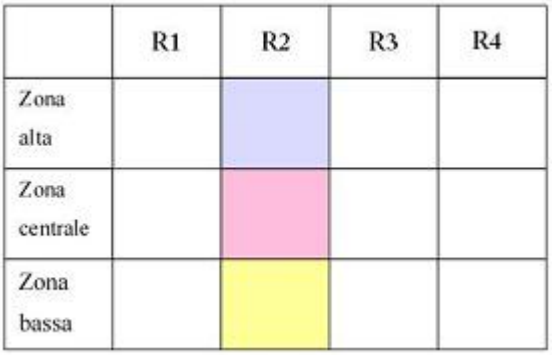



(Figura : Confronto di test realizzati sul primo prototipo a varie temperature)

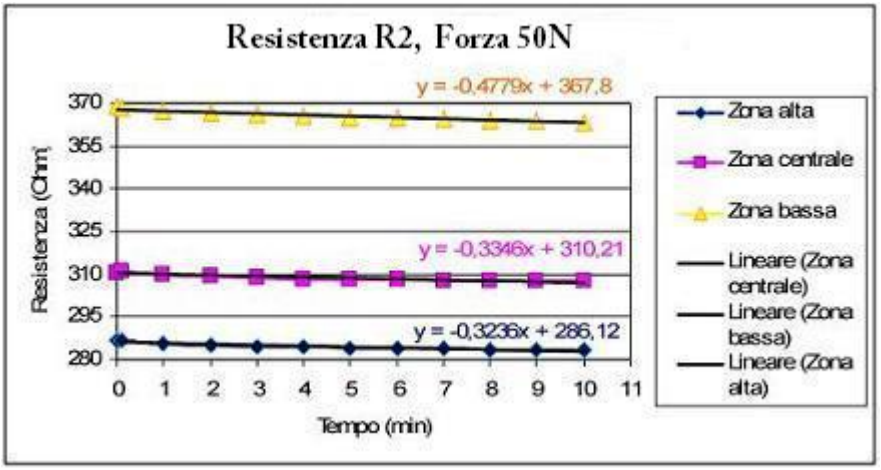
Questo è visibile anche nel grafico $\Delta R$ in funzione della pressione; si ha un massimo di $\Delta R$ pari a 0.004 M e un minimo di 0 M nel test a 10°C, mentre in quello 50°C il $\Delta R$ massimo è 0.03 e il minimo è 0.016. Si ha quindi un fattore almeno di un ordine di grandezza diverso nelle stesse condizioni di applicazione della pressione. Inoltre nel test effettuato a 50°C si tende ad avere un $\Delta R$ crescente all'aumentare della pressione applicata.

### 2.3.3 Variazione sensibilità - punto d'applicazione

Per questo esperimento è stato utilizzato il terzo prototipo (lo stesso per la tesi), si sono studiate e messe a confronto tre diverse zone di utilizzo di una pista conduttiva, utilizzando la tecnica dell'attuatore meccanico. In particolare, si riportano i grafici ottenuti a forza costante dalla pista conduttiva R2, al variare della posizione lungo la pista conduttiva, *zona alta*, *zona centrale* e *zona bassa*. La forza applicata è di 50N, corrispondente ad una spostamento del "dito meccanico" dall'alto verso il basso di circa 0.1 cm. Da questo grafico si può apprendere che il sensore rileva, anche se in modo poco evidente tale pressione. Inoltre cambiando i punti di applicazione sulla pista conduttiva, la risposta ottenuta non varia, è sempre la stessa.

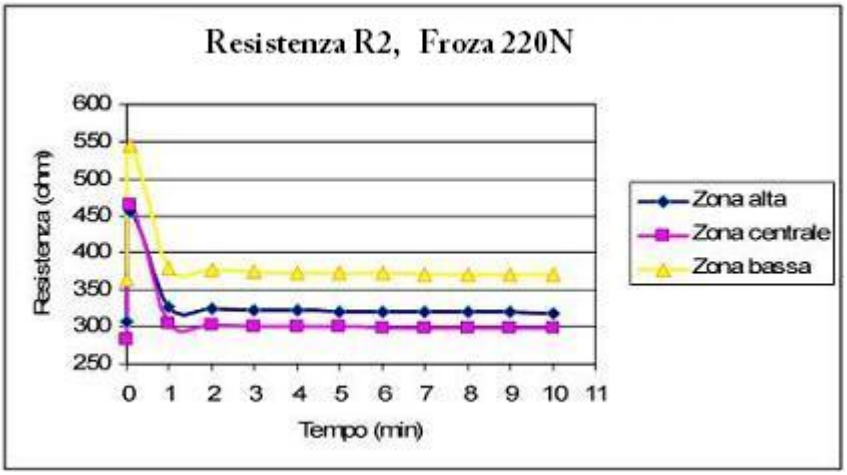



(Figura : Rappresentazione matriciale delle piste conduttive: le caselle colorate indicano la pista conduttiva dalla quale viene rilevata la resistenza, il colore delle singole caselle indica la zona specifica in cui è applicata la forza e corrisponde a quello usato nei grafici successivi. )



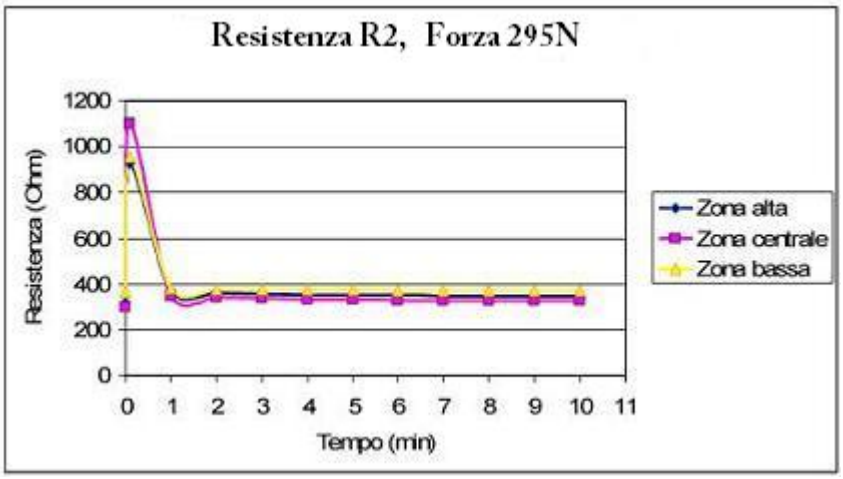

(Figura : Confronto di test effettuati sulle 3 zone della traccia R2 applicando una forza di 50N.)

Ugualmente è stato fatto applicando una forza maggiore, di 220N, corrispondente ad uno spostamento di 0.3 cm dell'attuatore meccanico. In questo grafico è visibile il picco, in corrispondenza della pressione, che va poi diminuendo abbastanza velocemente, per poi stabilizzarsi ad un certo valore dopo il primo minuto.




(Figura : Confronto di test effettuati sulle 3 zone della traccia R2 applicando una forza di 220N.)

E infine si è applicata la forza massima esercitabile dall'attuatore meccanico, di 295N, con discesa della testa operatrice di 0.5 cm, con il seguente risultato.



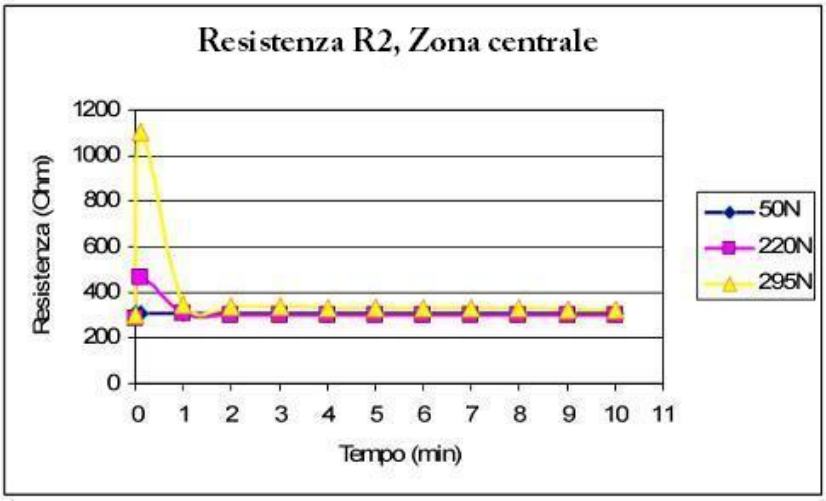

(Figura : Confronto di test effettuati sulle 3 zone della pista conduttiva R2 applicando una forza di 295N.)

### 2.3.4 Proporzionalità tra $\Delta$R e $\Delta$V

Con il grafico successivo, si vuole mettere in evidenza come, a seconda della pressione esercitata, si ottenga un segnale proporzionale; infatti applicando una forza di 50 N non è visibile una variazione evidente della resistenza. Mentre per forze di 220 N, si osserva un leggero picco, che aumenta con l'applicazione di forze pari a 295 N.

| | R1 | R2 | R3 | R4 |
|---|---|---|---|---|
| Zona alta | | | | |
| Zona centrale | | | | |
| Zona bassa | | | | |

(Figura : Rappresentazione matriciale delle piste conduttive: le caselle colorate indicano la pista dalla quale viene rilevata la resistenza, il colore giallo indica la zona specifica in cui è applicata la forza)



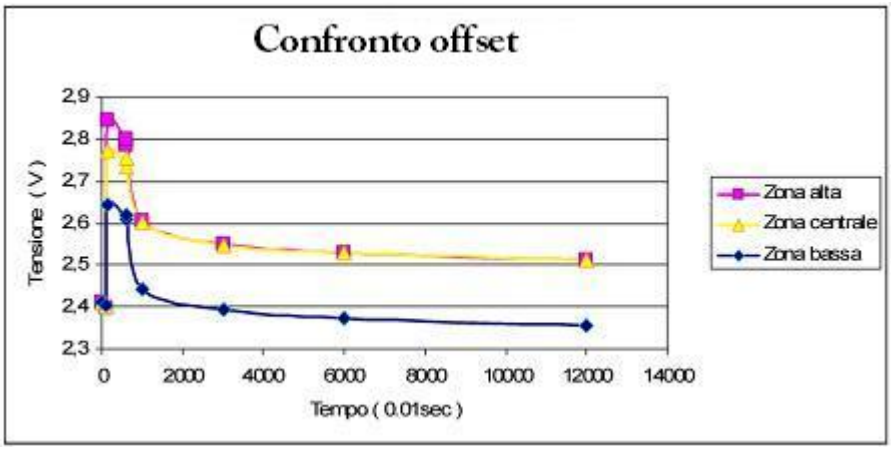

(Figura : Confronto di test effettuati sulla zona centrale della pista conduttiva R2 con applicazione di diverse forze)

Questi grafici sono stati effettuati per ogni zona della medesima pista conduttiva e di tutte le piste conduttive del sensore.

Anche in questo caso si sono analizzate tutte le piste conduttive e le tre sezioni della pista conduttiva al variare della forza applicata. Per ottenere un confronto migliore tra le varie zone della pista conduttiva, sottoposte alla stessa pressione, si è calcolato l'offset; si è fatto uno slittamento del livello del segnale elettrico rispetto ad un livello di riferimento, che in questo caso è stato il valore di riposo di una delle tre piste conduttive.

| | R1 | R2 | R3 | R4 |
|---|---|---|---|---|
| Zona alta | | | | |
| Zona centrale | | | | |
| Zona bassa | | | | |

(Figura : Rappresentazione matriciale delle piste conduttive: le caselle colorate indicano la pista dalla quale viene rilevata la resistenza, il colore delle singole caselle indica la zona specifica in cui è applicata la forza e corrisponde a quello usato nei grafico successivo)



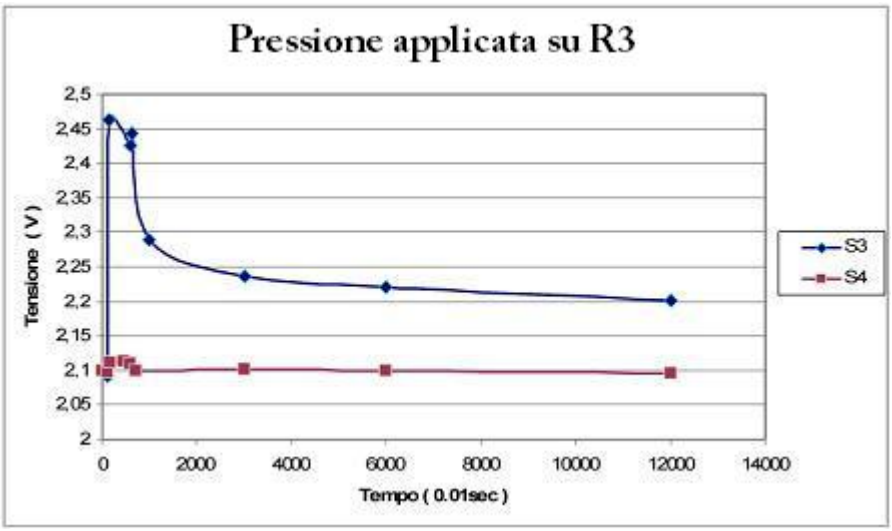

(Figura : Confronto di test effettuati sulle 3 zone della pista conduttiva R3, applicando una forza di 295N, utilizzando nel grafico i punti caratteristici della curva)

### 2.3.5 Influenza della pressione tra tracce adiacenti

Inoltre si è sperimentato che se viene applicata la pressione ad una pista conduttiva, quella vicina ne risente molto poco. Infatti, perché il sensore rilevi in maniera evidente la pressione, questa deve essere applicata alla pista conduttiva stessa. Questo è dimostrato nel grafico successivo, dove la pressione è esercitata sulla pista conduttiva R3 e la tensione viene rilevata sia sulla pista conduttiva R3, che su quella accanto, la pista conduttiva R4.

| | R1 | R2 | R3 | R4 |
|---|---|---|---|---|
| Zona alta | | | | |
| Zona centrale | | | | |
| Zona bassa | | | | |

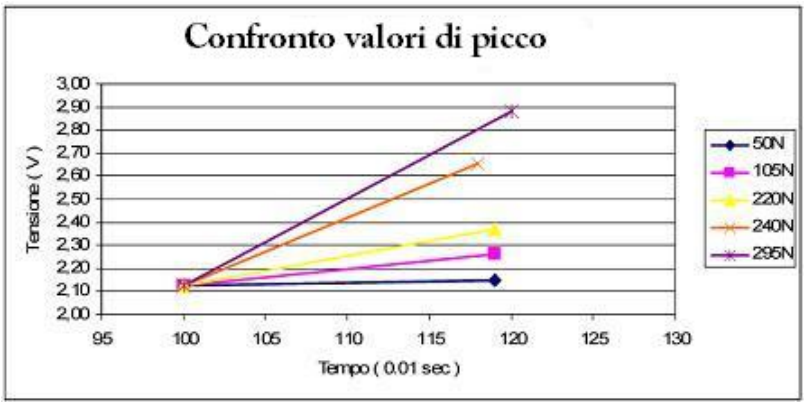
(Figura : Rappresentazione matriciale delle piste conduttive: le caselle colorate in rosa indicano la pista dalla quale viene rilevata la resistenza, in azzurro la pista sulla quale è applicata la forza )




(Figura : Confronto di test in cui si è applicata una pressione nota alla traccia R3, rilevamento sia della resistenza in R3 sia la resistenza della pista conduttiva vicina, R4)

### 2.3.6 Velocità di risposta del sensore

E' stata analizzata la velocità di risposta del sensore all'applicazione della pressione pari a 0.2 secondi, e la proporzionalità del valore della tensione, misurata in corrispondenza al picco con la pressione. Si può vedere come all'aumentare della forza applicata, aumenti il corrispondente valore della tensione.




(Figura : Confronto valori delle tensioni ottenute con forze sempre maggiori e tempo di risposta del sensore)

### 2.3.7 Aspettative teoriche sul funzionamento del sensore

Di seguiti sono riportate le conclusioni alle quali siamo giusti dopo un'analisi dei dati precedentemente illustrati:

Di seguito sono riportati in modo compatto tutti i risultati dei test eseguiti precedentemente, per avere un'idea chiara di quali sia il comportamento da aspettarci dal sensore tattile, durante la fase di test degli applicativi:

1. Il valore delle piste resistive del sensore cambia dopo aver subito una pressione, questo perché il silicone ha un tempo di isteresi dopo la pressione (deformazione), cioè una volta esercitata una pressione, il silicone si deforma (si allunga la pista resistiva di $\Delta$l), prima di tornare alla lunghezza originale impiega un certo tempo. Se in questo tempo subisce una pressione il suo allungamento sarà inferiore di $\Delta$l e quindi il segnale inviato sarà più basso, anche se si è applicata la stessa forza.

2. Il temperatura del sensore influisce sul suo funzionamento, in particolare con temperature basse ( 10°), la variazione di resistenza è molto piccola, mantre per temperature più alte (50°) la variazione è molto più accentuata (circa un ordine di grandezza in più). Quindi possiamo dire che con l'aumentare della temperatura aumenta anche la risposta alle sollecitazioni. Probabilmente con temperature basse il silicone è meno elastico, quindi il $\Delta$l massimo e molto piccolo.

3. In media, in una traccia conduttiva, la zona più sensibile alla pressione (cioè il $\Delta$R più elevato) è la *zona bassa* per ogni traccia. Poi segue la *zona media*, ed in fine la *zona alta*. Un motivo potrebbe essere la presenza dei contatti elettrici con i fili che collegato il connettore 9 poli con la traccia. Agendo vicino il punto di contatto si modifica non solo la lunghezza della pista, ma anche il collegamento con essa. Un altra motivazione potrebbe risiedere durante il processo di realizzazione del prototipo, in una drogaggio non omogeneo della pista conduttiva.

4. C'è proporzionalità tra il $\Delta$R e il $\Delta$V quando si esegue una pressione, ma non tra $\Delta$R e $\Delta$P, cioè non sempre ad una stessa pressione corrisponde una stessa variazione di resistenza.

5. Le traccie adiacenti ad una traccia che subisce una pressione, risentono molto poco di tale pressione, perché la variazione di resistenza avviene quando la pressione è esercitata direttamente sulla pista conduttiva, modificando la sua lunghezza. In questo modo si riesce ad identificare bene la zona di applicazione della forza sul sensore.

6. La velocità di risposta del sensore è all'incirca di 0.2 secondi, con piccole variazioni di 0.02 secondi dovuta al valore della forza applicata. Quindi campionare a 50Hz (0.02 secondi) per eliminare i disturbi del segnale, è una scelta ragionevole.

# Capitolo 3

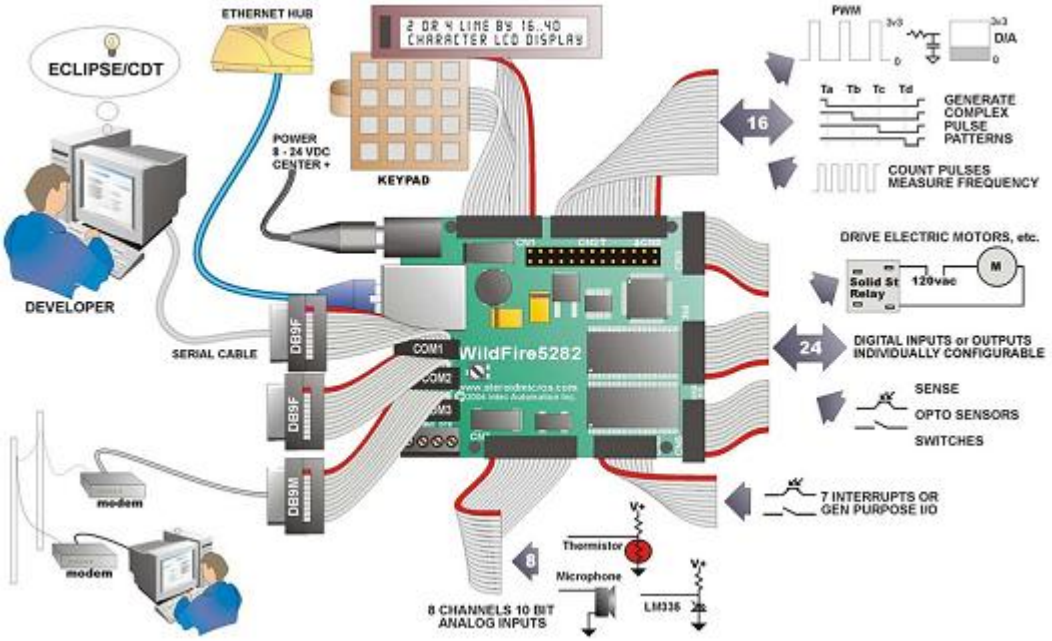
# La scheda WildFire 5282

La scheda WilFire 5282 [Inc04] rappresenta il "ponte" tra l'hardware del sensore ed il software sviluppato, si occupa di "campionare" il segnale proveniente dall'interfaccia del sensore, ed inviarlo al software attraverso la scheda Ethernet integrata in essa. In realtà questo componente è stato scelto non solo per la funzionalità di convertitore A/D, ma perché offre un tool di sviluppo abbastanza completo, ed inoltre presenta caratteristiche hardware molto interessanti per un ulteriori sviluppi[1].
Di seguito sono presentate le caratteristiche hardware della scheda, i tool necessari per sviluppare software, e la configurazione della scheda per le funzionalità Ethernet.

[1]Vedi sezione: 7 a pagina:98

## 3.1 Caratteristiche dell'hardware

La scheda può svolgere un grande numero di funzionalità, riassunte nella figura che segue:




(Figura : Schema delle diverse funzionalità offerte)

Vediamo meglio l'elenco delle caratteristiche della scheda:

- **Microprocessor**: 64MHz FreeScale Integrated ColdFire Version 2 Microcontroller.
- **Memoria**: 512K fast on-chip FLASH EPROM, 64K fast on-chip SRAM, 2 - 4 MB flash per memorizzare, 16 MB fast SDRAM per l'esecuzione di programmi.
- **SD Card Socket**: lettura di schede SD removibili con capacità 16 MB - 1GB.
- **Timer Port (2x10 header)**: per il controllo di moroti passa-passo, e conversione D/A.
- **Interrupt Port (2x5 header)**: per rispondere ad eventi critici.

- **A/D Port (2x8 header)**: 8 input analogici programmabili, con risoluzione di 10 bit e frequenza massima di campionamenti di 140KHz
- **3 porte di I/O configurabili (3−2x5 headers**)
- **Debug Port**
- **Ethernet Port (RJ45 Socket)**: velocità 10/100Mb, half o full duplex.
- **3 porte seriali (3-2x5 headers)**: 3 UARTs con RS232 e livelli di segnale: 2 DCE + 1 DTE.
- **CAN 2.0B Port (screw terminal)**
- **LCD/Keypad Port (2x7 header)**
- **Additional Timers**: 4 timer per la gestione periodica di eventi
- **Clock/Calendar**
- **Alimentazione**: 6 - 24 vdc unregulated input - 2.1mm center, Switching power supply on-board;

Le caratteristiche principali utilizzate per il lavoro di tesi sono, la funzionalità di convertitore analogico/digitale, e la possibilità di usare la rete ethernet per inviare i dati così campionati. In particolare, per quanto riguarda la frequenza di campionamento, è stata fatta un analisi preliminare sull'interfaccia del sensore, per verificare la presenza di disturbi nel segnale. Il disturbo principale riscontrato è dato dai 50Hz della rete elettrica, per eliminare ciò occorre campionare il segnale proprio sulla frequenza da eliminare. In realtà il segnale viene campionato al doppio di essa (100Hz), sarà poi il software a fare la correzione sui 50Hz, questo perché il disturbo è dovuto principalmente all'hardware dell'interfaccia. Se si migliorasse l'hardware dell'interfaccia si potrebbe campionare con una frequenza maggiore, cioè diminuire il tempo di risposta ed avere maggiori informazioni sul segnale generato dal sensore. Si può vedere dalle caratteristiche tecniche che la precisione per la conversione A/D è di 10 bit, precisione sufficiente per i nostri scopi, il range di tensione della porta A/D va da 0 a 5V, che mappati sui 10 bit si ha una precisione all'circa di 5mv (in realtà è poco inferiore ad essa).
Per quanto riguarda la comunicazione via ethernet è stato scelto di utilizzare il protocollo UDP, sia per ragioni di performance, sia perché la scheda è collegata direttamente con il pc, quindi si ha una perdita di pacchetti molto bassa (quasi nulla). Ai fini di questa tesi, l'utilizzo del protocollo UDP è

un compromesso accettabile, visto che il lavoro svolto non ha come obiettivo inserire la scheda in un contesto dove si richieda l'uso di un protocollo come il TCP. In futuro, se sarà necessario modificare questo aspetto, occorre semplicemente riscrivere parte del programma per la scheda wildfire, ed il modulo software che si occupa della ricezione dei dati da essa (DataSkin.java[2]).
In seguito sono riportati i tool di sviluppo forniti con la scheda:

- SBCTool Editor, una versione di Eclipse modificata ad-hoc per la connessione con la scheda.
- Compilatore C per il microcontrollore.
- Runtime Libraries in liguaggio C.
- Demo di utilizzo di alcune funzioni di libreria.

## 3.2 Configurazione e programmazione della scheda

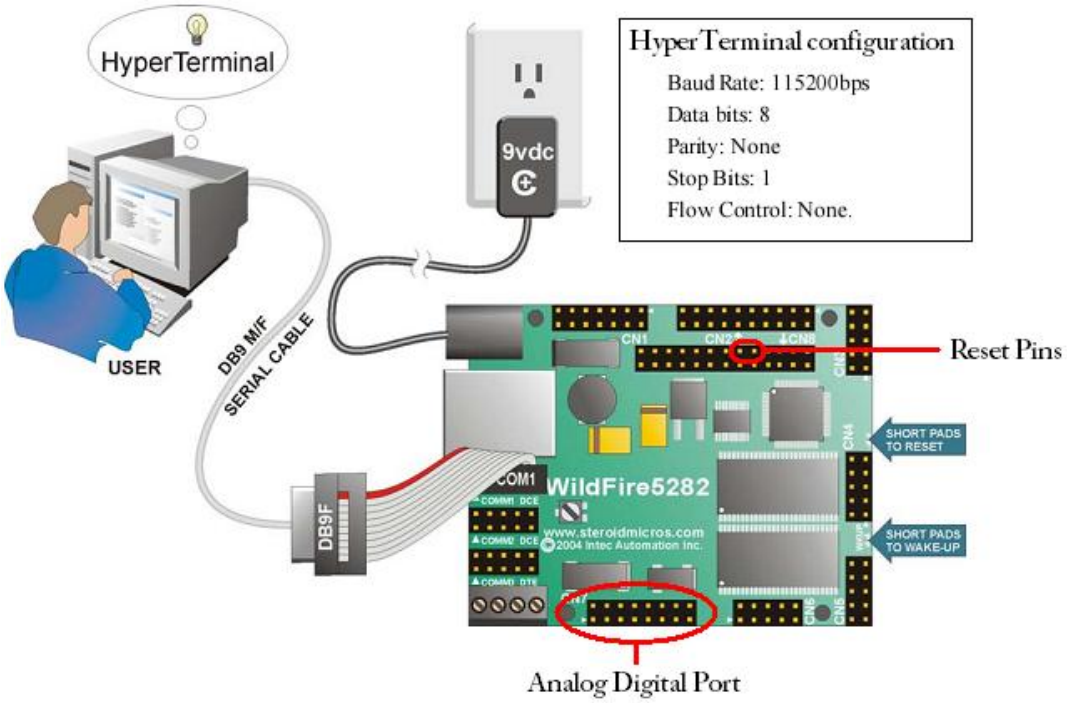
Prima di descrivere la parte di programmazione, è necessario configurare sia le impostazioni di rete della scheda, sia l'autorun dei programmi. Per far ciò è possibile utilizzare SBCTool, ed eseguire il comando "*set attributo valore*" nella schermata "SBC View" del tool. La configurazione utilizzata per la WildFire è la seguente:

```
base:  16
baud:  115200
autorun:  off
watchdog:  off
time:  26/11/2007-06:14:01
server:  192.168.0.1
client:  192.168.0.2
gateway:  192.168.0.1
netmask:  255.255.255.0
dns:  0.0.0.0
filename:  ConvAD.s19
filetype:  S-Record
ethaddr:  00:CF:52:82:CF:01
```

[2]Vedi sezione 5.3.2 a pagina 59

Per quanto riguarda la programmazione è stato utilizzato lo stesso programma "SCBTool" per compilare ed ottenere il file "*.s19", il quale sarà salvato sul microcontrollore e mandato in esecuzione.

Per quanto riguarda la scrittura del software sulla scheda, è possibile sia scrivere nella memoria SRAM del controllore (per fare i test), sia memorizzare il programma sulla memoria SD aggiunta come espansione. In entrambi i casi, non è stato utilizzato il l'SBCTool, perchè si sono riscontrati di versi problemi nell'invio dei dati (forse dovuti ai driver installati per la porta USB-Serial utilizzata per la comunicazione). *É* stato utilizzato l'HyperTerminal di Windows XP, sia per la comunicazione con la seriale, sia per la comunicazione ethernet, vediamo come:




(Figura : Collegameto PC - WildFire utilizzato)

**Scrittura in memoria SD esterna** il programma è memorizzato in modo permanente, si ha la possibilità di eseguirlo automaticamente all'accensione della WIldFire, occorre semplicemente impostare il campo autorun ad on (comando: "`set autorun on`") dall' HyperTerminal, e lasciare i due piedini di reset (evidenziati in figura) non collegati. Per

eseguire la scrittura su SD occorre utilizzare l'HyperTerminal con il comando per la scrittura da porta seriale: "~dfl sd", dopo di che andare nella barra dei menu di HyperTerminal e selezionare "Trasferimento ⇒ Invia file" e selezionare il file "∗.s19" prodotto dopo la compilazione con SBCTool. Lo svantaggio della scrittura su SD è quello di utilizzare la porta seriale, ovvero un banda di comunicazione molto bassa.

**Scrittura nella SRAM del microcontrollore** In questo caso, una volta eseguito il reset della scheda (ad esempio staccando l'alimentazione), il programma verra perso. *É* possibile salvare i dati nella memoria SRAM sia utilizzando la porta seriale, sia utilizzando la scheda ehternet (comando: "~dn <indirizzo PC> <Nome file>.s19")). In quest'ultimo caso, si può sfruttare la banda di comunicazione offerta dalla rete LAN 10/100 per inviare i dati a velocità superiori rispetto alla porta seriale. Questo tipo di comunicazione risulta molto utile, soprattutto durante la fase di sviluppo e test del programma.

## 3.3 Programma sviluppato

Il programma sviluppato per la funzionalità descritte nel paragrafo precedente è il seguente:

```
IL LATEX FA LE BIZZE CON LE TABULAZIONE
#include <wf5282.h> // Le RTL per la WildFire
#include <stdio.h>
#include <stdlib.h>
#include <opentcp.h>
#define BUF_LEN 2048
char buf[BUF_LEN];
//Questa funzione è chiamata soltanto dal TCP/IP subsystem durante la
chiamata di tcp_tick.  int network_handler(char handle, char event, int ip,
short port, short buffidx, short datalen) {
switch (event) {
case UDP_EVENT_DATA:
RECEIVE_NETWORK_BUF(buf, datalen);
buf[datalen] = 0; printf(%s/n, buf); break;
default:  printf(Unknown UDP event.); } return 0; }
int main(int argc, char** argv) {
```

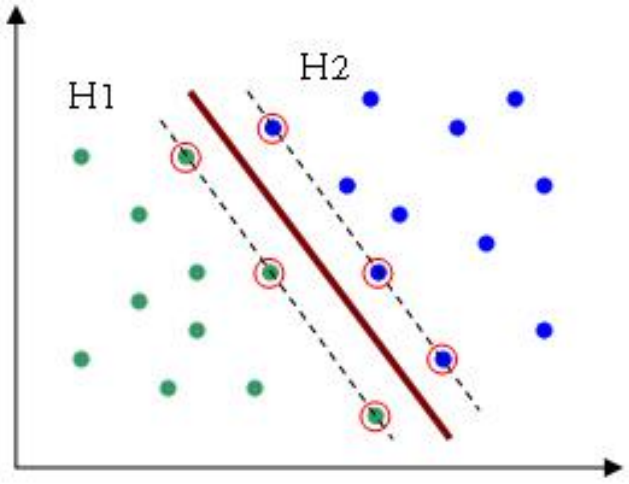
```
int i, inputKey, res; tcpip_init(0); printf( Address:
%d.%d.%d.%d/n,IP1(localmachine.localip), IP2(localmachine.localip),
IP3(localmachine.localip), IP4(localmachine.localip) );
unsigned int addr = make_ip(192, 168, 0, 1);
unsigned short port = 6543; //Porta di comunicazione utilizzata
char udpsock = udp_getsocket(0, network_handler, UDP_OPT_SEND_CS |
UDP_OPT_CHECK_CS); //Protocollo
char udphandle = udp_open(udpsock, port); //Creazione connessione UDP e
collegamento con handle
int* data = (int*)OTCP_TXBUF+UDP_APP_OFFSET; //impostazione dimensione del
pacchetto
ad_init( AD_SAMPLE_4TICS, AD_8CHAN_1SCAN ); //Campionamento a 2microSec
while( 1 ) {
delay( 9.998 );//Impostazione della frequenza di campionamento
tcp_tick();
for( i = 0; i < 8; i++ ) data[i] = mv_get(ad_get( i));//Valori convertiti in
volt
//Invio del pacchetto UDP cosi configurato res = udp_send(udpsock, addr,
port, (void*)data, NETWORK_TX_BUFFER_SIZE-UDP_APP_OFFSET, sizeof(int)*8);
printf(
if(chkch())//Terminazione del programma { inputKey = getch();
if(inputKey == 'q') { printf(QUIT /n); break; } } } return 0; }
```

(CHIEDERE se: Riportare la descrizione delle funzioni di libreria utilizzate per effettuare il campionamento e l'invio dei dati su UDP.)

# Capitolo 4

# Il classificatore SVM

Le Support Vector Machines ( abbreviate in SVM), sono un insieme di metodi di apprendimento supervisionato per la regressione e la classificazione di pattern, sviluppati negli anni '90 da Vladimir Vapnik ed il suo team presso i laboratori Bell AT&T [CL01]. Le SVM sono dei classificatore binari, in grado di apprendere come separare elementi appartenenti a due classi differenti. L'idea di base, è quella di trovare un iperpiano separatore, che massimizzi la distanza tra gli elementi delle due classi, su cui si fa apprendimento. I **Support Vector** sono proprio i punti appartenenti al training set, che definiscono l'iperpiano separatore (i punti più vicini ad esso).




(Figura : Esempio di separazione lineare tra due classi H1 e H2, gli elementi cerchiati in rosso sono i Suppor Vector.)

Una delle caratteristiche interessanti delle SVM, è la capacità di riuscire a dividere sia insiemi di dati linearmente separabili, sia non linearmente sep-

arabili. Infatti, quando non è possibile individuare un iperpiano separatore, si ha la possibilità di proiettare i punti del piano, in un altro piano con una dimensione maggiore, dove i punti sono separabili. In altre parole, quello che si fa, è utilizzare una funzione che mappa gli elementi di uno spazio $x$ in un altro $F(x)$, dove risultano linearmente separabili. Per definire una funzione di mappatura degli elementi, occorre calcolare il prodotto scalare fra i trasformanti dei Support Vector, ed il vettore d'ingresso: $F(x_s)^T F(x)$. In pratica è sufficiente definire una funzione kernel $K(x, x)$, che calcola tale prodotto scalare tra i punti trasformati. Si tratta quindi di scegliere funzioni "kernel", che rispettino la proprietà di definire un prodotto scalare fra due elementi, e che riescano a definire una mappatura che renda il problema linearmente separabile, nel nuovo spazio.

## 4.1 Fondamenti teorici

Dato l'insieme di esempi con elementi preclassificati $\{(x_1, y_1) \dots (x_p, y_p)\}$, dove ogni $x_i$ rappresenta un elemento del training set, ed $y_i$ rappresenta la classe di appartenenza di tale elemento, tale insieme è possibile esprimerlo come:

$$\{(x_1, y_1)..(x_p, y_p)\} \quad x_i \in \Re^N \quad y_i \in -1, +1$$

Mentre possiamo modellare il classificatore binario come:

$$\{f_\lambda(x) : \lambda \in \Lambda\} \qquad f_\lambda(x) : \Re^N \longrightarrow \{-1, +1\}$$

dove le $f_\lambda(x)$ sono chiamate ipotesi (o anche funzioni di decisione, tale funzione indica a quale delle due classi appartiene l'elemento x), e $\Lambda$ è l'insieme dei parametri che caratterizzano un particolare classificatore. L'insieme $f_\lambda(x) : \lambda \in \Lambda$ viene chiamato **spazio delle ipotesi** indicato con H, la dimensione di tale spazio viene chiamata "dimensione di Vapnik-Chervonenkis" [Vap98, Vap95] (indicata anche con VC). La dimensione di VC, è l'indice legato alla capacità di generalizzazione di una classe di classificatori.
Per costruire un buon classificatore occorre minimizzare *l'errore teorico* $R(\lambda)$ (chiamato anche *actual risk*).

$$R(\lambda) = \int |f_\lambda(x) - y_i| \, P(x, y) dx dy \quad (1)$$

In generale la distribuzione $P(x, y)$ è sconosciuta, quindi non è possibile utilizzare $R(\lambda)$, questo perchè supponiamo che il training set sia si stato generato a partire da una distribuzione sconosciuta $P(x, y)$. Tuttavia è possibile ricavare un'approssimazione di $P(x, y)$ dal training set stesso, e calcolare un'approssimazione del actual risk, chimata *errore empirico*:

$$R_{emp}(\lambda) = \frac{1}{l} \sum |f_\lambda(x_i) - y_i| \quad (2)$$

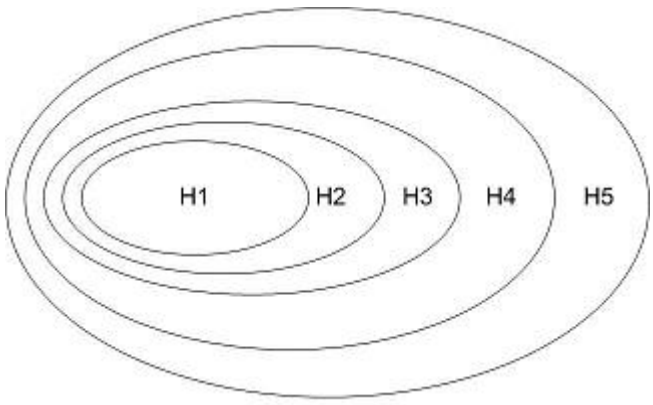
Il modo in cui mettere in relazione l'errore teorico con l'errore empirico è dovuto a Vapnik e Chervonenkis[CL01], che dimostrarono la seguente disequazione:

$$R(\lambda) \leq R_{emp}(\lambda) + \sqrt{\frac{h(ln\frac{2l}{h}) - ln\frac{\eta}{4}}{l}} \quad (3)$$

con $0 \leq \eta \leq 1$, $h$ è la dimensione di VC di $f_\lambda$; tendenzialmente un $h$ più alto implica un classificatore con maggiore capacità di generalizzazione, ma non necessariamente migliore in assoluto rispetto ad un altro. Il termine composto da tutta la radice quadrata, viene comunemente definito *VC-confidence*. Per ottenere l'errore teorico minimo, bisogna minimizzare sia l'errore empirico sia il rapporto tra la dimensione VC e il numero di punti $\frac{h}{l}$, occorre perciò trovare un punto di trade-off tra VC-confidence ed empirical risk, ovvero un valore di $h$ ottimo (spesso legato al numero di parametri liberi del classificatore), per ottenere buone prestazioni. Per trovare il trade-off migliore seguiamo il principio di "*Minimizzazione del Rischio Strutturale*" (Structural Risk Minimization - SRM), in tale principio lo spazio delle ipotesi viene diviso in sottoinsiemi concentrici:

$$H1 \subset H2 \subset \ldots \subset Hn \subset \ldots$$

con la proprietà che $h(n) \leq h(n+1)$ dove $h(n)$ è la dimensione VC dell'insieme Hn.




(Figura : Sottoinsiemi concentrici)

Nella ricerca della migliore f, l'SRM suggerisce di risolvere il seguente problema:

$$\min_{H_n} \left( R_{emp}(\lambda) + \sqrt{\frac{h(n)}{l}} \right) \quad (4)$$

In questo modo, il problema è ben formulato però si presentano subito due inconvenienti:

1. La dimensione VC di Hn potrebbe essere difficile da calcolare.
2. Anche assumendo che la dimensione VC sia calcolata per Hn, non è semplice risolvere il problema di minimizzazione. In molti casi prima si cercherà di minimizzare il rischio empirico per ogni Hn, e quindi si sceglierà quello specifico Hn che minimizza l'intera equazione.

Il controllo della dimensione VC di una tecnica di apprendimento durante l'addestramento, non è semplice. Le SVM realizzano questo obiettivo, minimizzando un limite sulla dimensione VC e sul numero di errori di training allo stesso tempo.

## 4.2 Classificatore lineare su dati linearmente separabili

Esaminiamo il caso più semplice di utilizzo delle Support Vector Machines. In questo caso il training set è definito come $\{\vec{x_i}, y_i\}, i = 1 \dots l, x_i \in \Re^d, y_i \in \{-1, +1\}$ dove i punti sono linearmente separabili da un iperpiano separatore. I punti che appartengono all'iperpiano soddisfano l'equazione: $\vec{w}\vec{x} + b = 0$, dove $\vec{w}$ è un vettore perpendicolare ad un iperpiano, $\frac{b}{\left\|\vec{w}\right\|}$ è la distanza dell'iperpiano dall'origine, e $\left\|\vec{w}\right\|$ è la norma euclidea di $\vec{w}$.

(Figura : Iperpiano separatore (w,b) per il training set)

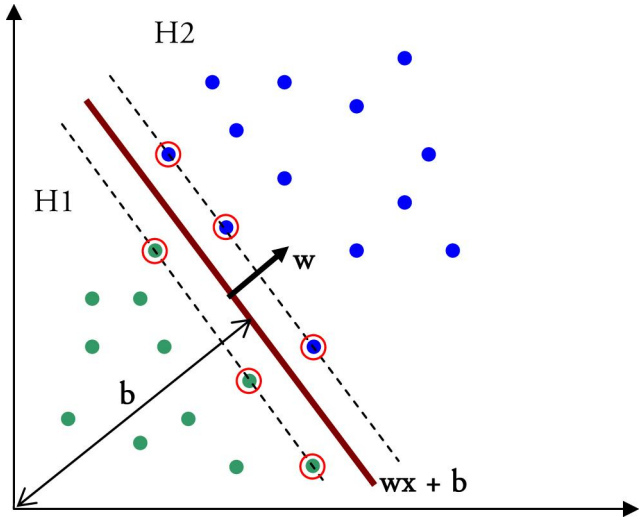
Ogni elemento del training soddisfa uno dei seguenti vincoli:

$$\vec{w}\vec{x_i} + b \geq +1 \quad y_i = +1 \quad (5)$$
$$\vec{w}\vec{x_i} + b \leq -1 \quad y_i = -1 \quad (6)$$

Si può anche riscrivere come:

$$y_i \left[ \vec{w}\vec{x_i} + b \right] \geq +1 \quad i = 1 \dots l \quad (7)$$

In questo caso lo spazio delle ipotesi considerato potrà scriversi come:

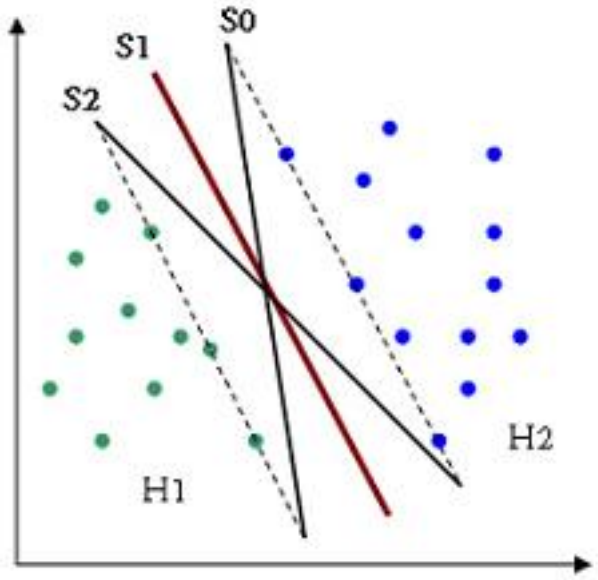



$$f_{w,b} = sign\,(w * x + b) \qquad (8)$$

dove la funzione *sign* è un discriminatore binario.
Per poter minimizzare l'actual risk dovremo allora mantenere il più piccola possibile la norma $\|w\|$ che riduce la VC-dimension, e allo stesso tempo, dovremo scegliere *(w,b)* che soddisfino $y_i \left[\vec{w}\vec{x_i} + b\right] \geq +1 \; i = 1 \ldots l$ per minimizzare l'empirical risk. Dunque, l'algoritmo SVM ci permette di trovare proprio la coppia *(w,b)* che minimizza l'actual risk risolvendo il problema di ottimizzazione così modellato:

$$min \tfrac{1}{2} \|w\|^2 \qquad (9)$$
$$y_i \,(w * x_i + b) \geq 1 \quad i = 1 \ldots l \qquad (10)$$

Un iperpiano che soddisfi questa equazione sarà detto iperpiano separatore ottimo o anche a massimo margine, dove per margine si intende la distanza che c'è tra le due regioni delle classi, misurata sulla perpendicolare all'iperpiano. Che un tale iperpiano abbia un maggior potere di generalizzazione lo si note molto bene nella figura che segue, in cui gli iperpiani S0 e S2 separano correttamente le due classi ma non sono a massimo margine, al contrario dell'iperpiano S1 il quale è a massimo margine.




(Figura : Gli iperpiani S0, S1 e S2 compiono l'ottimizzazione perfetta così che il rischio empirico viene annullato. Comunque, S0 risulta essere l'iperpiano ottimo perché massimizza il margine)

Per risolvere il problema di minimo vincolato, di solito si ricorre alla teoria dei moltiplicatori di Lagrange [Ber82], che consente di ottenere una funzione di decisione del tipo:

$$L(w, b, \Lambda) = \frac{1}{2} \|w\|^2 - \sum_{i=1}^{l} [y_i (w \cdot x + b) - 1] \quad (11)$$

in cui $\Lambda = (\lambda_1 \dots \lambda_l)$ è il vettore dei moltiplicatori di Lagrange non negativi, relativi ai vincoli impostati nella (10). La lagrangiana deve essere minimizzata rispetto a w e b e contemporaneamente massimizzata rispetto a $\Lambda \geq 0$.
Alla fine di ciò, l'iperpiano ottimo può essere scritto come una combinazione lineare dei vettori del training set:

$$w^* = \sum_i \lambda_i^* y_i x_i = \Lambda^* y x \quad (12)$$
$$f(x) = w^* x + b = \Lambda^* y x \cdot x + b \quad (13)$$

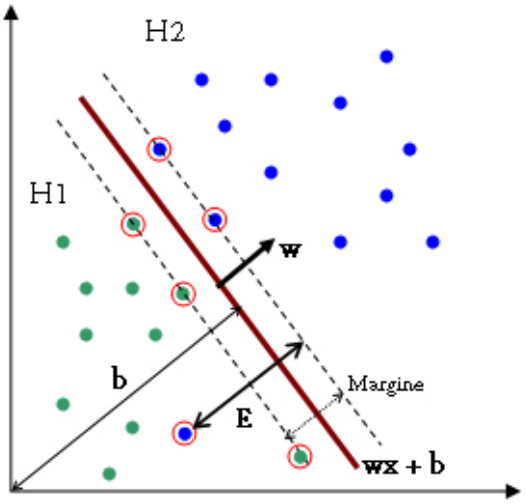
La funzione così ottenuta è una funzione quadratica. Attraverso altri passi di semplificazione (il duale) sulla funzione si arriva ad ottenere un classificatore della forma:

$$f(x) = sign\left(\sum_{i=1}^{l} y_i \lambda_i^* (x^* x_i) + b^*\right) \quad \forall x_i \quad (14)$$

Nella soluzione, tutti i punti $x_i$ per cui il corrispondente moltiplicatore $\lambda_i$ è strettamente maggiore di zero, vengono detti **Support Vector**, e si trovano su uno dei due margini con l'iperpiano. Tutti gli altri punti del training set hanno il corrispondente $\lambda_i$ uguale a zero, e non influenzano il classificatore.

In sostanza, i *Suppor Vector* sono i punti critici del training set, sono i più vicini all'iperpiano di separazione; se tutti gli altri punti venissero spostati o rimossi senza oltrepassare i margini definiti sull'iperpiano, e l'algoritmo di apprendimento venisse ripetuto, darebbe esattamente lo stesso risultato.

## 4.3 Classificatore lineare su dati non linearmente separabili




(Figura : Insieme di punti non linearmente separabili)

In questo caso, non è possibile individuare nessun iperpiano separatore per i punti di questo training set, quindi l'algoritmo precedente non determina nessuna soluzione ammissibile. Per risolvere il problema è sufficiente rilassare i vincoli (5)(6) introducendo una margine di errore $\xi$. A questo punto, i vincoli possono essere riscritti come:

$$\vec{w}\vec{x_i} + b \geq +1 - \xi_i \quad y_i = +1 \quad (15)$$
$$\vec{w}\vec{x_i} + b \leq -1 + \xi_i \quad y_i = -1 \quad (16)$$

Quindi per ogni elemento è stato aggiunto un margine di errore, ottenendo $\sum_i \xi_i$ come limite superiore al numero massimo di errori. In formato più compatto i vincoli si possono riesprimere come:

$$y_i \left[ \left( \vec{w}\vec{x_i} \right) + b \right] - 1 + \xi_i \geq 0 \qquad (17)$$

Assegnando un peso agli errori $\xi$, si può cambiare la funzione da minimizzare in:

$$\begin{gathered} min \frac{1}{2} \left\| w \right\|^2 + C \left( \sum_i \xi_i \right)^k \qquad (18) \\ \xi_i \geq 0 \quad \forall i = 1 \ldots l \end{gathered}$$

dove C e k sono parametri da impostare per definire il costo della violazione dei vincoli. Minimizzare il primo termine corrisponde a minimizzare la VC-confidence, mentre minimizzare il secondo termine corrisponde a minimizzare l'empirical risk. Applicando la tecnica del rilassamento lagrangiano, e calcolando il duale si ottiene:

$$\begin{gathered} max F(\Lambda) = \Lambda \cdot 1 - \frac{1}{2} \Lambda \cdot D\Lambda \\ \Lambda \cdot y = 0 \\ \Lambda \geq 0, \Lambda \leq C \end{gathered}$$

Il classificatore è dato da:

$$f(x) = sign \left( \sum_{i=1}^{l} y_i \lambda_i^* (x * x_i) + b^* \right) \qquad \forall x_i \qquad (14)$$

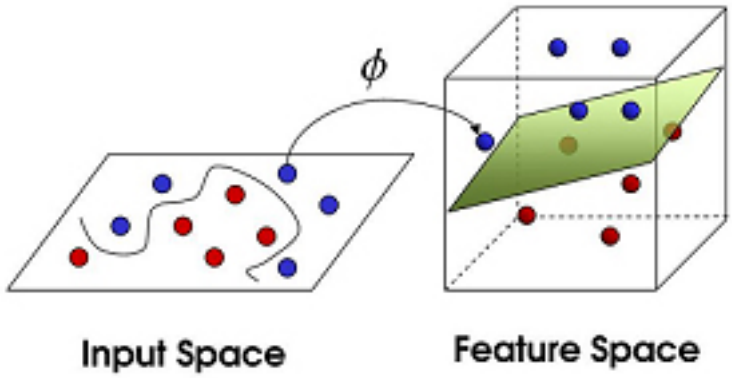
La differenza rispetto al caso linearmente separabile consiste nel fatto che i moltiplicatori lagrangiani $\lambda_i$ sono limitati superiormente da C.

## 4.4 Classificatore non lineare

In realtà le superfici lineari (trattate fin ora), non sono appropriate per la risoluzione di molti problemi. In questi casi, per individuare un iperpiano separatore si può pensare di aumentare la dimensione H dello spazio del problema. In sostanza si può effettuare un mapping dei dati in uno spazio H a maggiore dimensionalità, denominato spazio di Hilbet, attraverso la funzione :

$$\phi : \Re_d \rightarrow H$$

(Figura : Mapping in uno spazio a dimensione superiore)




Sapendo che l'algoritmo di apprendimento dipende dai dati solo attraverso il loro prodotto scalare, per realizzare il mapping dei dati è sufficiente applicare $\phi$ ad ogni elemento, ovvero: $\phi\left(\vec{x_i}\right)\phi\left(\vec{x_j}\right)$. Uno spazio di dimensione maggiore causa però seri problemi di calcolo, perché l'algoritmo di apprendimento deve lavorare con vettori di grandi dimensioni. Per ovviare a questo problema si può introdurre una funzione *Kernel* che restituisce il prodotto delle immagini dei suoi due argomenti:

$$K\left(\vec{x_i}, \vec{x_j}\right) = \phi\left(\vec{x_i}\right)\phi\left(\vec{x_j}\right)$$

nell'algoritmo di training sarà sufficiente utilizzare K, senza rendere esplicita la funzione $\phi$. Sostituendo nell'algoritmo di apprendimento, tutte le occorrenze di $\vec{x_i}\vec{x_j}$ con $K\left(\vec{x_i}, \vec{x_j}\right)$ si ottiene il nuovo classificatore:

$$f(x) = sign\left(\sum_{i=1}^{l} y_i \lambda_i^* \left(K\left(\vec{x_i}, \vec{x_j}\right) + b^*\right)\right) \qquad \forall x_i \qquad (2.)$$

Per individuare per quali Kernel esiste una coppia $\{H, \phi\}$ con le proprietà descritte precedentemente, è possibile utilizzare il teorema di Mercer: esiste una funzione $\phi$ e un'espansione $K(u,v) = \sum_{k=1}^{\infty} a_k \phi_k(u)\phi_k(v)$ con $a_k$ coefficienti positivi se e solo se, per ogni funzione $g \neq 0$ tale che:

$$\int g^2(u)\,du \leq +\infty$$

si ha che:

$$\int K(u,v)\,g(u)\,g(v)\,dudv \geq 0$$

Questa è sempre soddisfatta per potenze positive del prodotto scalare $K\left(\vec{u},\vec{v}\right)=\left(\vec{u}\cdot\vec{v}\right)^{P}$.
La funzione Kernel va scelta accuratamente in base al tipo di problema, è sempre possibile mappare l'input su uno spazio di dimensione maggiore del numero di punti del training set, per produrre un classificatore prefetto. Tuttavia questo generalizzerebbe molto male su dati nuovi, per via dell'overfitting.
I tipi di Kernel comunemente usati sono i seguenti:

- Lineare $K\left(x,y\right)=x\cdot y$
- Polinomiale $K\left(x,y\right)=\left(1+x\cdot y\right)^{d}$
- Radial Basis function $K\left(x,y\right)=exp\left(-\gamma\|x-y\|^{2}\right)$
- Gaussian Radial Basis function $K\left(x,y\right)=exp\left(-\frac{(x-y)^{2}}{2\vartheta^{2}}\right)$
- Multi-Layer Perceptron $K\left(x,y\right)=tanh\left(b\left(x\cdot y\right)-c\right)$

## 4.5 Librerie disponibili

Per quanto riguarda le librerie disponibili che implementano le SVM, sono state analizzate le $SVM^{light}$ e LIBSVM, ed è stato scelto di utilizzare LIBSVM, perché più aggiornata, più semplice nell'utilizzo, ed offre più tool attraverso i quali è possibile testare singolarmente le diverse configurazioni SVM.

### 4.5.1 $SVM^{Light}$

$SVM^{light}$ [Joa98] è un'implementazione efficiente sviluppata in linguaggio C delle Support Vector Machine di Vapnik, per il problema della classificazione, della regressione e dell'apprendimento di una funzione di ranking . Il tool è rilasciato freeware alla pagina web *http://svmlight.joachims.org/* ed costituito da due moduli: svm learn e svm classify. Il primo modulo è utilizzato per addestrare l'SVM attraverso un training set, mentre il secondo per testare l'SVM attraverso un test set. Dopo la fase di apprendimento, nel caso della classificazione, il tool restituisce la classe corrispondente ad ogni esempio di test. La classificazione avviene per due classi distinte: la seguente versione del tool non permette la multilabel classification. Per quest'ultima classificazione e per la gestione di output strutturati, esiste un'estensione del tool

chiamata SVMstruct della quale si parlerà successivamente. Oltre alla classificazione con approccio induttivo, il tool permette una classificazione di tipo transduttivo. Nell'approccio induttivo (dal particolare al generale) il learner cerca di indurre la funzione decisionale (il generale) cercando di minimizzare l'error rate sulla sola distribuzione dei pattern d'esempio (il particolare). In diverse situazioni non ci si preoccupa della particolare funzione decisionale, ma ci si preoccupa della classificazione di un set di esempi (il test set) con il minimo errore possibile. Questo è l'obiettivo dell'approccio transduttivo (dal particolare al particolare). Nel caso della regressione, l'output Y appartiene ai numeri reali. Il tool permette anche l'apprendimento di una ranking function. Una ranking function può essere utilizzata ad esempio nell'information retrieval per stilare una classifica delle pagine web che soddisfano una certa query Q.
L'apprendimento avviene attraverso il seguente comando:

*svm learn [-options] train file model file*

dove il file di training contiene gli esempi di addestramento, mentre model file è il modello che verr´a creato. Il tool supporta diverse opzioni che riguardano l'apprendimento. Ricordiamo ora le pi´u importanti mentre per le altre si rimanda alla documentazione ufficiale. L'opzione -z permette di scegliere il problema sul quale si vuole lavorare: è possibile scegliere tra classificazione (c), regressione (r) e ranking (p). L'opzione -c, permette di determinare il trade-off tra l'errore e il margine. Ad un alto valore di C, dove C ¿ 0, corrisponde un'alta penalità dovuta agli errori, mentre un piccolo valore di C, incrementerà gli errori di training. Attraverso l'opzione -t è possibile utilizzare diversi tipi di kernel, cioè:

- Lineare
- Polinomiale
- Radial Basis Function
- Percettrone Multistrato
- Kernel definito dall'utente

Il test avviene tramite il seguente comando:

svm classify [-options] test file model file predictions file

Il file d'output delle SVMlight è costituito da una linea per ogni campione di test contenente il valore della funzione decisionale per l'esempio considerato. Per il problema della classificazione, il segno della label predetta determina la classe. Nel problema della regressione, la label è il valore predetto, mentre per il ranking è usata per ordinare gli esempi di test.

### 4.5.2 *LIBSVM*

LIBSVM [LIBSVM] consiste in un tool per la classificazione (C-SVC,nu-SVC), regressione (epsilon-SVR, nu-SVR) e stima della distribuzione (SVM ad una classe). Il tool a differenza di SVMlight supporta la classificazione multiclasse ed è rilasciato freeware alla pagina *http://www.csie.ntu.edu.tw/cjlin/libsvm/*. Un obiettivo importante di questa libreria è quello di permettere agli utenti, non solo informatici, un facile utilizzo delle SVM. Il package è costituito da tre tool a linea di comando: svm-train, svm-predict ed svm-scale. Il primo modulo ´e utilizzato per addestrare l'SVM attraverso un training set, il secondo per testare l'SVM attraverso un test set, il terzo modulo è utilizzato per effettuare una normalizzazione dei dati nel range definito dall'utente, che può essere ad esempio [1,-1] oppure [0, 1].Il tool è in grado di risolvere efficacemente cinque problemi:

- C-Support Vector Classification
- v-Support Vector Classification
- $\xi$-Support Vector Regression
- v-Support Vector Regression
- Distribution Estimation (one-class SVM)

I passi fondamentali nell'utilizzo del tool sono i seguenti:

1. Convertire i dati in input nel formato del tool che si intende utilizzare.
2. Effettuare la normalizzazione dei dati nel range opportuno ( es. [1,-1] )
3. Scegliere il tipo di funzione kernel da utilizzare, ad esempio Kernel Gaussiano
4. Utilizzare la cross-validation per determinare i parametri C (soft-margin) e $\frac{3}{4}$ (ampiezza della gaussiana)

5. Con i valori di C e $\frac{3}{4}$ trovati con la cross-validation, ri-effettuare la fase di training

Normalizzazione Supponendo di aver eseguito precedentemente la fase di preprocessing, quindi di avere il data set nel formato adottato dal tool, è possibile passare alla fase di normalizzazione dei dati tramite il comando svm-scale. L'obiettivo di questa fase è quello di effettuare dei piccoli aggiustamenti ai dati allo scopo di renderli più adatti alle valutazioni del kernel. Supponendo ad esempio di prendere in considerazione un Kernel Gaussiano - un particolare tipo di kernel che fa parte di quelle funzioni kernel chiamate RBF (Radial Basis Functions). Se una certa feature di un campione, presenta un maggiore range di variazione rispetto ad un'altra, allora questa dominerà la sommatoria nella gaussiana, mentre feature con piccole variazioni di range saranno essenzialmente ignorate. Da questo se ne può ricavare che feature con maggiore range di variazione, avranno maggiore attenzione da parte dell'algoritmo SVM . *É* necessaria quindi una normalizzazione dei valori delle feature in modo tale che questi cadano all'interno dello stesso range, che può essere ad esempio [0,1] oppure [-1,+1] . Questa normalizzazione pu´o essere eseguita attraverso il comando svm-scale che accetta in input un data set e restituisce come output lo stesso data set normalizzato nel range scelto e un insieme di parametri di scalatura. Questi dati riscalati verranno poi utilizzati dal modulo che realizza l'apprendimento. Il formato del comando di scalatura è il seguente:

*svm - scale - s param scalatura training file ¿ training file scalato*

Per specificare il range di scala, vengono utilizzate due opzioni che specificano rispettivamente il lower bound e l'upper bound del range, ovvero "-l" e "-u".

Addestramento e Test Come si può notare la cross validation non utilizza tutti gli esempi di training, infatti ad ogni iterazione alcuni di questi vengono esclusi dalla valutazione. *É* necessario quindi effettuare l'addestramento sull'intero training set utilizzando i parametri che son stati determinati dalla precedente fase. La fase di training avviene tramite il comando svm-train nel seguente modo:

*svm - train [-options] training file scalato model file*

Il tool fornisce diverse opzioni di training. Per una descrizione più accurata si rimanda alla documentazione ufficiale. Per prima cosa il tool permette di scegliere quale tipo di problema risolvere, quindi quale tipo di SVM utilizzare, attraverso l'opzione "-s". *É* possibile scegliere tra:

- C-SVC
- v-SVC
- one-class SVM
- $\xi$-SVM
- v-SVR

Per quanto riguarda i kernel utilizzabili, quelli elencati sono quelli messi a disposizione dal tool. Di default è impostato il Kernel RBF.

- Lineare
- Polinomiale
- Radial Basis Function
- Sigmoidale

Una volta teminata la fase di training, si ha a disposizione la funzione decisionale appresa memorizzata nel model file e i dati di test a loro volta scalati. A questo punto è possibile effettuare le predizioni delle label per gli esempi di test attraverso il comando svm-predict che ha la seguente forma:

*svm - predict [-options] test file scalato model file predictions*

Come è possibile notare, l'input del comando consiste nel test set scalato, nel model file ed il file in cui memorizzare l'output. L'output consiste nelle predizioni delle label, $sgn\left(f\left(x\right)\right)$, del test-set.

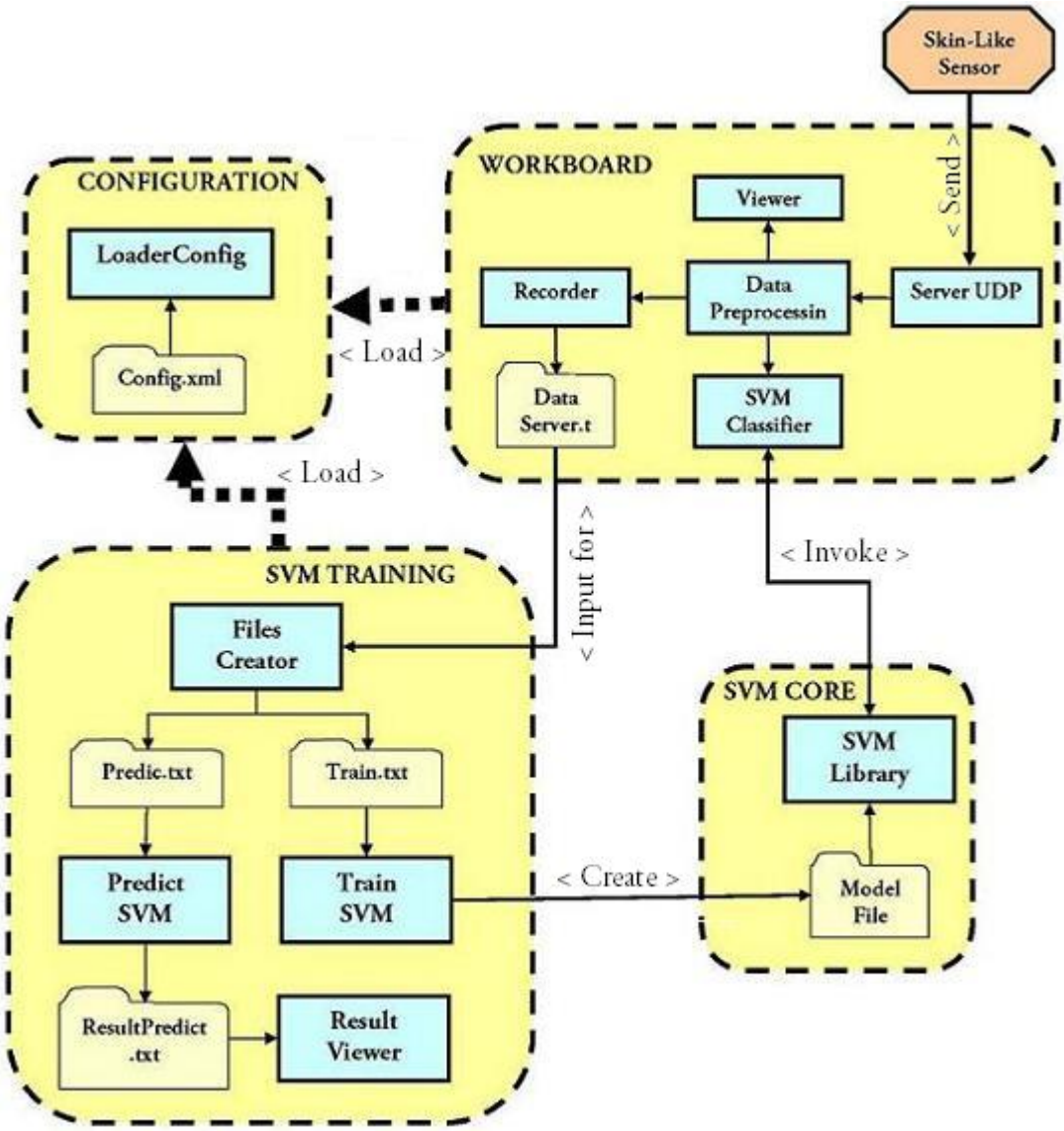
# Capitolo 5

# Tools sviluppati

In questo capitolo sarà anlizzato il software sviluppato per elaborare, classificare e visualizzare i dati in arrivo dal sensore tattile. Esso è formato da quattro tool distinti, uno per l'addestramento delle Support Vector Machines, un tool per elaborare i dati ricevuti dal sensore, effettuando particolari operazioni come: la derivata, la media esponenziale, amplificare o diminuire il segnale ed il salvataggio dei dati. Un tool di editor dei dati salvati, in sostanza permette di assegnare le etichette hai valori ricevuti dal sensore. E l'ultimo tool per impostare le configurazioni degli altri, per quanto riguarda i path delle directory, ed i valori di default per alcune funzionalità offerte. Inoltre sono riportate le screenshoot delle diverse interfacce grafiche realizzate per i tool.
Tutto il software è stato sviluppato in Java per due motivi principali: il primo è la portabilità offerta dal linguaggio, il secondo sono le librerie incluse nella piattaforma, sia grafiche che per la lettura di file XML, le quali hanno velocizzato l'implementazione dei diversi tool.

## 5.1 Struttura generale

La struttura generale del software è composta da *quattro sezioni*, ognuna delle quali occupa un posto preciso nella fase di sviluppo del classificatore. Vediamo la struttura:




(Figura : Struttura generale del software)

Nello schema disegnato, i moduli in ogni sezione rappresentano le funzionalità presenti in essa, non l'organizzazione delle classi. Inoltre sono presenti

degli elementi il cui disegno è una "cartella", questi non sono moduli software ma file di testo generati come output da alcuni moduli (come il Recorder). Questi file saranno successivamente l'input di altri moduli. La scelta di generare file di testo invece di passare direttamente l'input al modulo successivo, permette di analizzare passo passo il lavoro fatto, e di effettuare copie dei dati ottenuti fino a quel punto. Questo modo di procedere, sarà molto utile nella fase di test, infatti è possibile testare singolarmente ogni tipo di interazione da riconoscere, ottenuto un classificatore per essa (ovviamente si cerca il migliore), si salva il file di train con il quale è stato generato. Fatto ciò per tutte le tipologie di iterazioni da classificare, è possibile fare una "marge" di tutti i file di train precedentemente salvati, ed infine testare il classificatore così ottenuto.
Il processo per la costruzione di un buon classificatore, segue le seguenti quattro fasi:

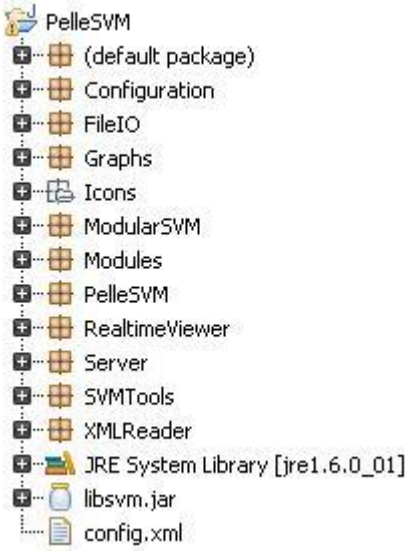
1. *Creazione di una board*: Attraverso il tool "ModularWorkBoard" è possibile creare una board che elabora i dati (ad esempio eliminando i rumori del segnale), li visualizza e permette il salvataggio in un file (per defalut è chiamato "DatiServer.txt").

2. *Classificazione dei dati*: Una volta salvati i dati, si esegue il tool "DataEditor", il quale visualizza i dati contenuti nel file "DatiServer.txt", e permette di classificare ogni dato (ovvero una 8-tupla di valori interi) con una etichetta che corrisponde alla classe di appartenenza.

3. *Addestramento della SVM*: A questo punto i dati presenti nel file "DatiServer.txt" vengono divisi su due file (come due insiemi disgiunti), train.txt e predict.txt. Il file train.txt sarà usato per l'addestramento dalla SVM, mentre per verificare la percentuale di successo si utilizzano i dati presenti nel file predict.txt. *É* possibile scegliere come dosare i dati nei rispettivi file, per dafault il 70% viene inserito nel file train.txt ed il restante nel file predict.txt. Al termine del training si sono creati i file "≺nome≻.model", che sono il risultato dell'addestramento e saranno utilizzati per la classificazione realtime.

4. *Classificazione real-time*: Nel tool "ModularWorkBoard" è presente un modulo che legge i file "≺nome≻.model" generati in precedenza, permette di utilizzare uno di essi per la classificazione dei dati in arrivo dal sensore, con la possibilità di variare il .model durante l'esecuzione. A questo punto ogni volta che viene ricevuto un dato, viene mostrata l'etichetta con la quale è stato classificato.

Di seguito viene riportata la lista di tutti i package prodotti, con una breve descrizione delle funzionalità offerte da ognuno di essi.




(Figura : Lista dei package sviluppati)

Questi package sono alla base dei quattro sezioni evidenziate nella struttura generale, ognuno dei quali può essere utilizzato anche in più moduli, come ad esempio il package "Configuration". Di seguito non viene presentato il diagramma delle classi per ogni package ma solo una loro descrizione sommaria. Questo perché tale diagramma sarà tracciato in seguito, quando saranno analizzati in profondità i tool che utilizzano questi package.
Di seguito segue la descrizione dei package:

- **Defalut Package**: in realtà non è un package ma è la directory di lavoro, essa contiene tutti i file "Main" delle applicazioni sviluppate. Inoltre sono disponibili le applicazioni di test, per verificare la correttezza delle funzionalità dei singoli tool.

- **Configuration**: lo scopo di questo package, è quello di leggere le configurazioni presenti nel file "config.xml", e metterle a disposizione dei diversi tool. Esso sarà analizzato meglio quando si parlerà del "Tool Configuration" (si seguito questa sezione).

- **FileIO**: offre le funzionalità di scrittura e lettura da file.

- **Graphs**: permette di visualizzare in un grafico i valori letti dal file "DatiServer.txt", "train.txt" e "predict.txt"; permettendo per ogni valore, di visualizzare la classe di appartenenza e di eliminarlo dal grafico.
- **Icons**: è la cartella contenente le icone usate dalle applicazioni.
- **ModularSVM**: implementa l'interfaccia grafica del applicativo "ModularWorkBoard"[1], con tutte funzionalità di una GUI (toolbar,caricamento e salvataggio della board, drag e cancellazione di moduli, etc..).
- **Modules**: in questo package sono implementati i moduli presenti nel tool ModularWorkBoard. Ogni classe del package per funzionare correttamente estende la superclasse Module.java, nella quale sono implemenetate le funzionalità di comunicazione con altri blocchi, pick-correlation, click-button e menu delle proprietà, facilitando la creazione di un nuovo modulo da aggiungere al software.
- **PelleSVM**: contiene l'interfaccia grafica per interagire con le funzionalità offerte dal package SVMTools.
- **SVMTools**: in esso sono presenti i file che si interfacciano con la libreria LIBSVM, implementano la funzionalità di training e predict di dati, e generando i ".model"[2], necessari per il classificatore.
- **XMLReader**: legge i dati presenti nel file "config.xml" e li memorizza nella classe "Const.java" nel package Configuration, rendendoli disponibili agli altri package impostandoli nelle variabili della classe "Const.java".

Nelle sezioni che seguono saranno analizzati i quattro moduli principali che compongono il software,

## 5.2 Tool: Configuration

Questo tool ha due funzionalità: la prima consiste nel contenere tutti i parametri (di default) di configurazione degli altri tool, come i path della directory di lavoro o dei file generati dai diversi tool. La seconda nel contenere le diverse configurazioni di SVM che saranno eseguite automaticamente nella fase di training[3] e di predict[4]. In questo modo aggiungere

[1] Vedi sezione 5.3 a a pagina 55
[2] "Vedi capitolo: 4 a pagina 32"
[3] Vedi sezione 5.5.1 a pagina 69
[4] Vedi sezione 5.5.2 a pagina **??**

nuove configurazioni è un'operazione molto semplice. Per fare ciò il tool è composto sia da una classe pubbica chiamata "Const.java", la quale contiene al suo interno solo variabili statiche, accessibili da tutte le classi. Sia da un file in formato XML nel quale sono contenute le configurazioni (nominate in precedenza), inoltre ad ogni campo del file XML corrisponde una variabile nel file "Const.java". In pratica ogni tool sviluppato, ha incluso nel proprio main le seguenti righe di codice:

```
ConfigLoader loader = new ConfigLoader();
    loader.loaderParamAndStamp(false);
```

con le quali si leggono i campi presenti nel file XML, e si impostano le corrispettive variabili presenti nella classe "Const.java". In questo modo si ha la possibilità di modificare alcune caratteristiche dei moduli, senza dover ricompilare il codice, è sufficiente un semplicemente un editor testuale. Inoltre ha facilitato l'inserimento di nuovi campi sia durante la fase di sviluppo del software, si nella fase di test del classificatore, per aggiungere nuove configurazioni per le Support Vector Machines. L'unico vincolo imposto è sul nome del file XML, ovvero "config.xml", ed inoltre tale file deve risiedere nella stessa directory della "≺main≻.class". Non è stata sviluppata nessuna interfaccia grafica tramite la quale si può modificare il file XML, perché non si è ritenuta necessaria, visto che basta un word processor come il blocco note per modificarlo.
Di seguito riporto il file "config.xml utilizzato nel tool, con una spiegazione dei campi che lo compongono:

```
<ConfigPelleSVM>

    <PathDir>
        <Dir> c://DatiPelle// </Dir>
    </PathDir>

    <DataFileServer>
        <Name> datiServer.txt </Name>
    </DataFileServer>

    <TrainFile>
        <Name> train.txt </Name>
    </TrainFile>

    <PredictFile>
        <Name> predict.txt </Name>
    </PredictFile>
```

```
<ListResult>
    <Name> resultTestPredict.txt </Name>
</ListResult>

<ParameterKernel>
    <Arg1> -s 0 -t 0 </Arg1>
    <Arg2> -s 0 -t 1 -d 1 </Arg2>
    <Arg3> -s 0 -t 1 -d 2 </Arg3>
    <Arg4> -s 0 -t 2 </Arg4>
    <Arg5> -s 0 -t 3 </Arg5>
    <Arg6> -s 0 -t 3 -r 1 </Arg6>
    <Arg7> -s 0 -t 3 -r 2 </Arg7>
    <Arg8> -s 0 -t 3 -r 2 </Arg8>
    <Arg9> -s 1 -t 0 </Arg9>
    <Arg10> -s 0 -t 1 -d 3 </Arg10>
    <Arg11> -s 0 -t 1 -d 4 </Arg11>
    <Arg12> -s 0 -t 1 -d 5 </Arg12>
    <Arg13> -s 0 -t 1 -d 6 </Arg13>
    <Arg14> -s 0 -t 1 -d 7 </Arg14>
    <Arg15> -s 0 -t 1 -d 8 </Arg15>
    <Arg16> -s 1 -t 2 -g 0.5 </Arg16>
    <Arg17> -s 1 -t 1 -d 1 </Arg17>
    <Arg18> -s 1 -t 1 -d 2 </Arg18>
    <Arg19> -s 1 -t 2 </Arg19>
    <Arg20> -s 1 -t 3 </Arg20>
    <Arg21> -s 1 -t 3 -r 1 </Arg21>
    <Arg22> -s 1 -t 3 -r 2 </Arg22>
    <Arg23> -s 1 -t 3 -r 2 </Arg23>
    <Arg24> -s 1 -t 2 -c 0.0001 </Arg24>
    <Arg25> -s 1 -t 2 -c 0.005 </Arg25>
    <Arg26> -s 1 -t 2 -c 0.001 </Arg26>
    <Arg27> -s 1 -t 2 -c 0.01 </Arg27>
    <Arg28> -s 1 -t 2 -c 0.1 </Arg28>
    <Arg29> -s 1 -t 1 -d 2 -g 5.1 </Arg29>
    <Arg30> -s 1 -t 1 -d 2 -g 5.1 -n 0.5 -e 0.11 </Arg30>
</ParameterKernel>
```

```
<ListOutputFile>
    <Arg1> C-SVClinear.model </Arg1>
    <Arg2> C-SVC-polynomial-d1.model </Arg2>
    <Arg3> C-SVC-polynomial-d2.model </Arg3>
    <Arg4> C-SVC-radialbasis.model </Arg4>
    <Arg5> C-SVC-sigmoid-c-r0.model </Arg5>
    <Arg6> C-SVC-sigmoid-c-r1.model </Arg6>
    <Arg7> C-SVC-sigmoid-c-r2.model </Arg7>
    <Arg8> C-SVC-sigmoid-c-r3.model </Arg8>
    <Arg9> nu-SVC-linear.model </Arg9>
    <Arg10> C-SVC-polynomial-d3.model </Arg10>
    <Arg11> C-SVC-polynomial-d4.model </Arg11>
    <Arg12> C-SVC-polynomial-d5.model </Arg12>
    <Arg13> C-SVC-polynomial-d6.model </Arg13>
    <Arg14> C-SVC-polynomial-d7.model </Arg14>
    <Arg15> C-SVC-polynomial-d8.model </Arg15>
    <Arg16> nu-SVC-radialbasis-g0.5.model </Arg16>
    <Arg17> nu-SVC-polynomial-d1.model </Arg17>
    <Arg18> nu-SVC-polynomial-d2.model </Arg18>
    <Arg19> nu-SVC-radialbasis.model </Arg19>
    <Arg20> nu-SVC-sigmoid-c-r0.model </Arg20>
    <Arg21> nu-SVC-sigmoid-c-r1.model </Arg21>
    <Arg22> nu-SVC-sigmoid-c-r2.model </Arg22>
    <Arg23> nu-SVC-sigmoid-c-r3.model </Arg23>
    <Arg24> nu-SVC-radialbasis-c-0.0001.model </Arg24>
    <Arg25> nu-SVC-radialbasis-c-0.005.model </Arg25>
    <Arg26> nu-SVC-radialbasis-c-0.001.model </Arg26>
    <Arg27> nu-SVC-radialbasis-c-0.01.model </Arg27>
    <Arg28> nu-SVC-radialbasis-c-0.1.model </Arg28>
    <Arg29> nu-SVC-polynomial-d2-g5.1.model </Arg29>
    <Arg30> nu-SVC-polynomial-d2-g5.1-n0.5-e0.11.model
</Arg30>
</ListOutputFile>

<GeneratedFile>
    <Name> predictGen.txt </Name>
</ GeneratedFile>

<KernelSelect>
    <Name> nu-SVC-radialbasis-c-0.001.model </ Name>
</ KernelSelect>
```

```
<NumInput>
    <Value> 8 </Value>
</NumInput>

<ScaleFactor>
    <Value> 1 </Value>
</ScaleFactor>

<ServerPort>
    <Value> 6543 </Value>
</ServerPort>

<BufferViewer>
    <Value> 50 </Value>
</BufferViewer>
```

Di seguito viene riportata una descrizione dei campi che compongono il file, ed i valori di default impostati per essi.

- **PathDir**: questo campo indica la directory in cui saranno salvati i file train.txt, predict.txt e tutti i *.model generati nella fase di training.
- **DataFileServer** : contiene in nome del file nel quale il modulo Recoder del tool ModularWorkBoard andrà a salvare i dati.
- **TrainFile**: il nome del file per i training.
- **PredictFile**: il nome del file sul quale si testa la percentuale di successo dopo la fase di training.
- **ListResult**: contiene i risultati della fase di predict per le diverse configurazioni SVM utilizzate.
- **ParameterKernel**: rappresenta la lista di configurazioni sulle quali si eseguirà il training e successivamente il predict.
- **ListOutputFile**: ogni campo corrisponde ad una configurazione in ParameterKernel, il nome rappresenta il tipo di configurazione utilizzata, il quale verrà visualizzato anche in seguito nel classificatore. Inoltre dopo aver eseguito la fase verifica delle diverse configurazioni, vengono generati file contenenti i risultati della fase di predict (questi file saranno in seguito utilizzati dalla classe ViewerResult).
- **GeneratedFile**: questo parametro è stato usato nella fase di sviluppo del sistema, corrisponde al nome di un file nel quale viengono generati casi specifici per verificare la classificazione fatta su di essi.

- **KernerSelected**: indica il kernel selezionato da default nel classificatore.
- **NumInput**: numero di interi contenuti in un dato inviato dal sensore. Questo parametro è presente perchè in questo modo si ha la possibilità di gestire sensori con un numero di resistenze differenti da quella utilizzata per il lavoro di tesi.
- **ScaleFactor**: utilizzato all'interno del filtro come valore di default per amplificare i dati ricevuti.
- **ServerPort**: porta si ascolto del server UDP.
- **BufferViewer**: numero di dati visibili nel "viewer module".

*É* bene sottolineare il rapporto "uno-a-uno" che c'è tra i campi di ListOutputFile e quelli in ParameterKernel, inoltre è possibile inserire un valore arbitrario per indicare il nome della configurazione (valori in ListOut).

## 5.3 Tool: ModularWorkBoard

L'obiettivo di questa applicazione è quello di fornire uno strumento semplice ed allo stesso tempo molto flessibile. Che permetta l'elaborazione dei dati in real time, la visualizzazione di tali elaborazioni, col fine di individuare un particolare preprocessing dei dati, in modo da renderli utilizzabili dal classificatore.
Per far ciò è stata scelta una struttura di composizione modulare, ovvero sono stati creati un insieme di moduli, dove ognuno di essi esegue una specifica funzione sui dati ricevuti. Inoltre implementano la stessa interfaccia, nella quale viene definito il metodo d'ingresso dei dati. In questo modo è possibile mettere in comunicazione due o più moduli, ed eseguire in serie delle funzioni per ogni dato. Ogni modulo è composto a sua volta da un "core", cioè la funzione eseguita sui dati d'ingresso. Ed un'interfaccia grafica, cioè l'icona che lo rappresenta (sulla board), attraverso la quale è possibile accedere alle proprietà del modulo, oppure collegare più moduli in serie tra loro.
Dopo un'analisi dei moduli da creare, è stata individuata la presenza in ognuno di essi di funzionalità in comune, come l'aggiunta e la cancellazione di un modulo dall'array di output[5], l'implementazione della pick correlation ed il menu delle proprietà. Quindi è stato scelto di sostituire l'implementazione

[5]Questo array rappresenta la lista dei moduli ai quali inviera il risultato dell'elaborazione.

dell'interfaccia con l'estensione di una classe dove tali funzionalità comuni a tutti i moduli sono implementate. In questo modo si sfruttano i vantaggi offerti dall'ereditarietà. La super-classe da estendere ha il nome di "Module" (nel package Modules), di seguito sono riportate le funzionalità implementate al suo interno:

- *Comunicazione tra moduli*: il metodo "*void inputCH(int[] input)*", tramite esso è possibile collegare l'output di un modulo con l'input di un altro.

- *Pick correlation*: ogni modulo ha tra le sue variabili un coppia di interi che indica la posizione nello schermo, ed un'altra coppia che indica la dimensione dell'immagine (da disegnare) relativa al modulo. Il metodo che implementa questa funzionalità ha la seguente firma: "boolean pickCorrelation(int x,int y)", in sostanza controlla, se i valor inseriti come parametri del metodo, sono all'interno di un rettangolo definito dalla posizione del module e la sua dimensione, ritornando un valore boolean che indica il risultato di tale confronto.

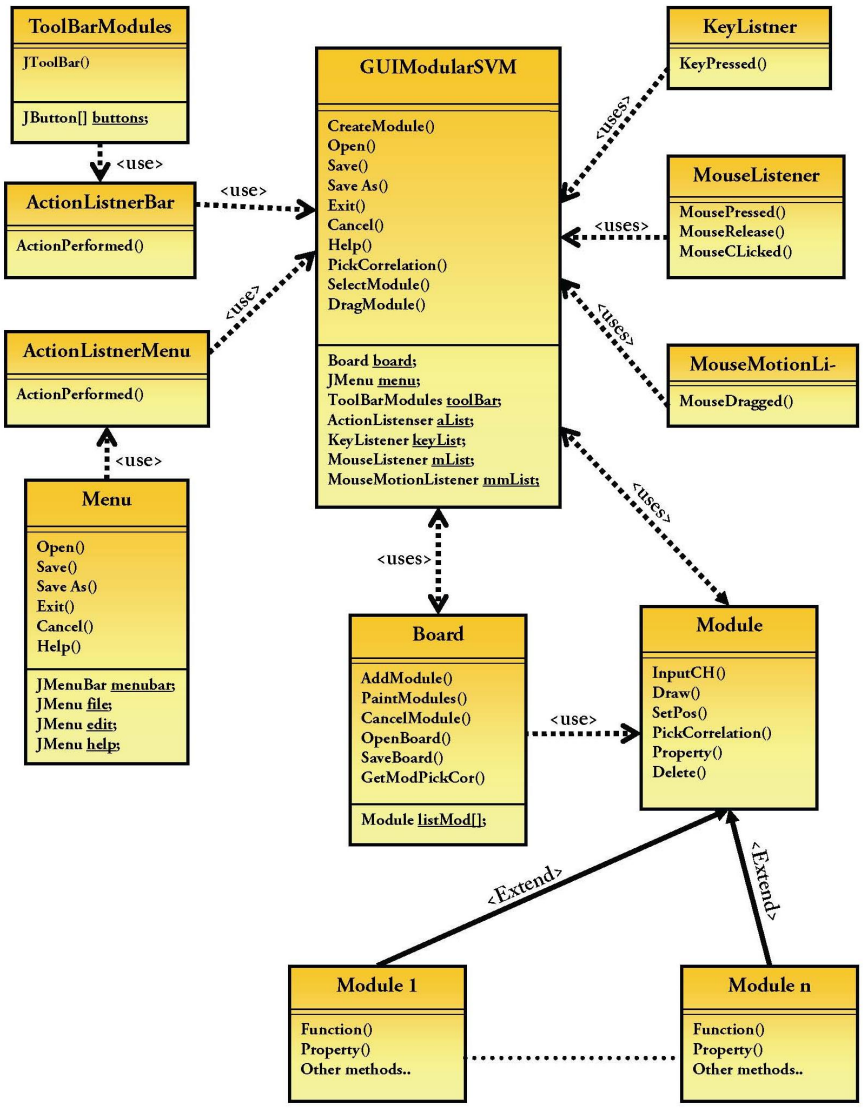
- *Aggiunta di un modulo in output*: la super-classe Module contiene al suo interno un array di moduli (variabile "Module[] listMod"), questo array rappresenta l'insieme di oggetti sui quali invocare il metodo "inputCH(int[] input)", ovvero "a chi passare i dati" dopo averli elaborati (se ciò è desiderato).

- *Cancellazione di un modulo dall'output*: in questo caso l'operazione è elementare, trovare il modulo e cancellarlo dalla lista, semplicemente sovrascrivendolo con l'ultimo elemento della lista (se lui non è l'ultimo), altrimenti cancellando il riferimento ad esso.

- *Caricamento dell'immagine*: cioè disegnare l'immagine che rappresenta il modulo nella board.

- *Menu per le proprietà*: viene creata una classe contenete la lista delle proprietà modificabili del moduli, inizialmente è vuota. Sarà delegato al modulo che estende la superclasse "Module" aggiungere oggetti (JComponent) relativi ai campi modificabili dall'utente.

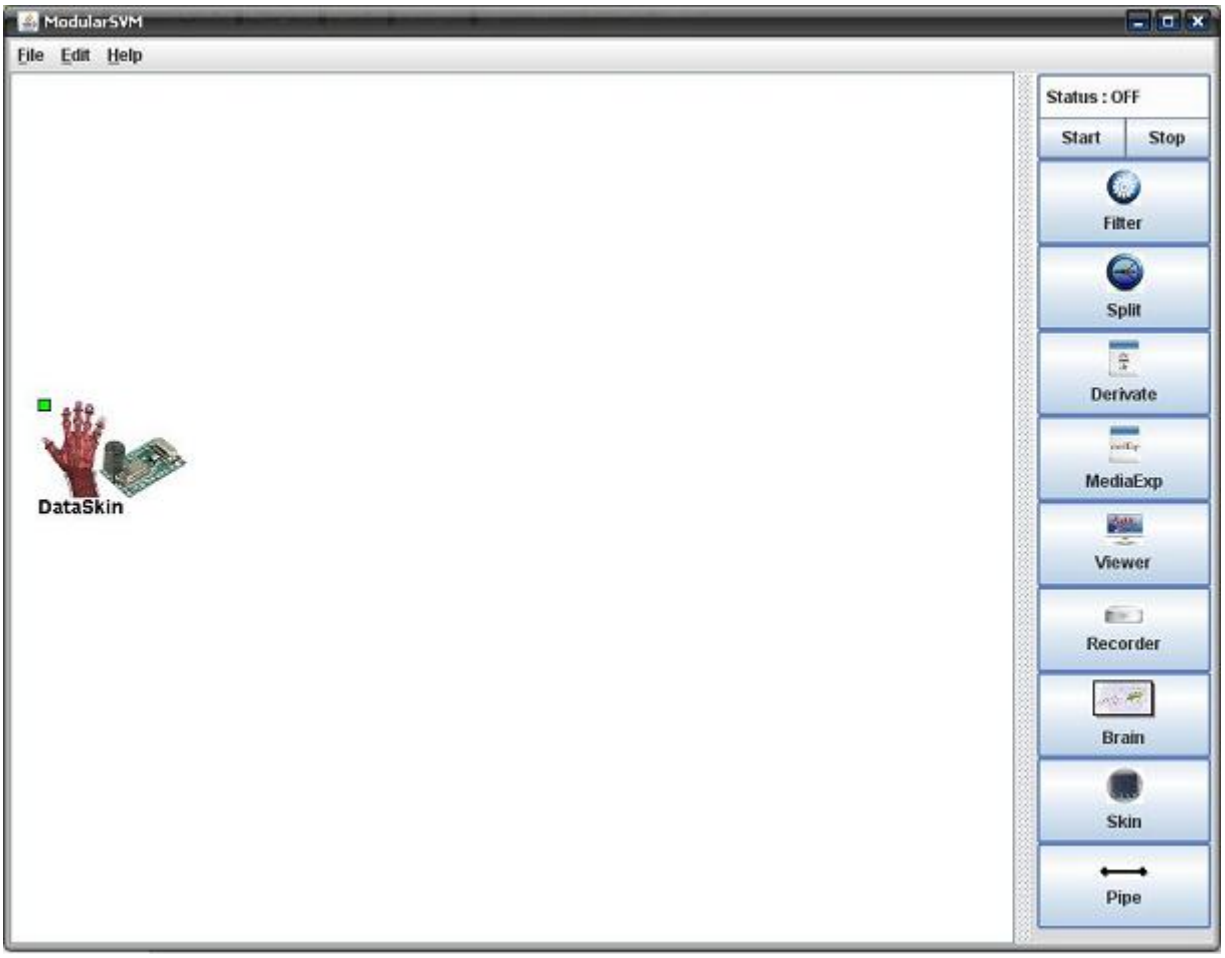
L'architettura finale del software è la seguente:




(Figura : Diagramma UML semplificato delle classi dell'applicazione)

### 5.3.1 Descrizione GUI del tool

Di seguito viene riportata una screenshoot dell'applicazione:




(Figura : Screenshoot dell'applicazione ModularWorkBoard)

L'interfaccia grafica si compone di una classe core chiamata "*GUIModularSVM*", che riceve gli eventi sia da tastiera che dal mouse, ogni evento ricevuto viene analizzato e gestito nel modo appropriato. In questa classe sono stati inseriti altre tre classi: Menu, ToolBarModules e Board. La classe "*Menu*" è posizionata nella barra dei file, in essa sono gestite le operazioni di caricamento, salvataggio di una Board, chiusura dell'applicazione, cancellazione modulo e una piccola guida che spiega come utilizzare il programma. Associato a tale menu c'è un Lister che riceve gli eventi generati dai

"sotto-menu" al suo interno, ed esegue il metodo appropriato nella classe "GUIModularSVM". Inoltre è stata attivata la funzionalità dei tasti rapidi, ovvero ad ogni voce del menu corrisponde una combinazione di tasti.
Già dal nome seconda classe "*ToolBarModules*" si intuisce il suo funzionamento, in essa sono stati inseriti un bottone per ogni modulo disponibile, ed è stato aggiunto un componente di "Status" che segnala se l'applicazione riceve o meno i dati dal sensore. La toolbar ha attiva la funzionalità di "fly", ovvero è possibile staccarla dall'applicazione e riposizionarla dove si vuole.
L'ultimo modulo "*Board*" rappresenta il tavolo di lavoro, al suo interno è presente la lista di tutti i componenti attivi (presenti nella board), memorizzati in un array di oggetti di tipo "Module". Esso gestisce le funzionalità di disegno dei moduli (attraverso l'utilizzo di una pila, per avere sempre in primo piano il modulo selezionato), cancellazione di un modulo, caricamento e salvataggio della board creata. Per default, quando si esegue l'applicazione viene aggiunto il componente "DataSkin", che non è presente nella toolBar, infatti esso è l'unico module che non è possibile cancellare.

### 5.3.2 I moduli funzione

I moduli creati sono stati sviluppati durante l'utilizzo del software, essi sono nati per rispondere alle esigenza di manipolazione dei dati ricevuti (ad esempio amplificandoli, oppure impostando un filtro sui valori) per poi visualizzarli. Per riuscire a distinguere visivamente quando si effettua una pressione sul sensore da quando il sensore non viene toccato. Sono stati implementati due tipologie di moduli, la prima che implementa sia un canale d'ingresso che d'uscita, la seconda che implementare solo un canale d'ingresso. Questo perché per alcuni moduli non ha senso inviare i dati, come ad esempio nel "Viewer", il quale semplicemente visualizza i dati ricevuti. Di seguito sono riportati i diversi moduli sviluppati durante il lavoro di tesi.

#### DataSkin

Questo modulo rappresenta il sensore tattile skin-like. Al suo interno è stato implementato un server UDP in ascolto su una specifica porta. La parte del server è stata implementata in un thread, nella classe:"ServerUDP". Il compito del server, è quello di ricevere i dati inviati dalla scheda WildFire 5282, e renderli accessibili a tutti gli altri moduli dell'applicativo. Va ricordato, che la WildFire 5282 invia pacchetti UDP contenenti i valori degli otto canali di conversione A/D (collegati nel nostro caso con l'interfaccia del sensore), tali valori non sono direttamente accessibili dal pacchetto UDP, occorre estrarli dal pacchetto attraverso un operazione di shift di bit e di conversione in deci-

male. Questo modulo è l'unico della board che non può essere ne "cancellato" ne "aggiunto", per cui viene aggiunto in ogni board creata, come operazione di default (dal software). L'icona del modulo è la seguente:

(Figura : Icona nella board del modulo DataSkin)

Nell'icona è presente in alto a sinistra un "led" verde , ogni volta che il server riceve un dato il led lampeggia, alternando il colore da verde a giallo. L'introduzione del led è dovuta principalmente per controllare il thread ServerUDP, perché è stata riscontrata una frequenza di ricezione dei dati non costante, oppure una sospensione di ricezione per qualche secondo. Dopo un analisi della situazione si è visto che il problema non risiede nella scheda ma proprio nel thread, che a volte rallenta (probabilmente dovuta alla sua implementazione).
L'unica proprietà visibile di questo oggetto (che l'utente può modificare), è la frequenza di acquisizione dei dati, si passa da un minino di 10 ms ad un massimo di 1 s. Un'ultima cosa da dire su questo modulo, è l'accessibilità ad alcuni sui metodi da parte del componente di "status", presente nella toolbar. In questo modo è possibile sospende il thread (attraverso il pulsante "STOP"), quando se ne ha bisogno.

### Pipe

Questo componente permette la comunicazione tra due moduli. A livello di composizione, una "pipe" collega solo due moduli tra loro, inoltre è stato previsto uno schema di connessioni della tipologia "uno-a-molti", ovvero ogni modulo può inviare dati a più moduli ma può ricevere solo da uno specifico[6]. L'icona della pipe è la seguente:

(Figura : DataSkinIco)

[6]In realtà, questa impostazione può essere facilmente cambiata a livello software, basta modificare una variabile booleana, ciò è stato previsto per futuri sviluppi.

Questo componente può essere visto come "un ponte" di comunicazione tra due moduli, in quale riceve i dati, non effettua nessuna elaborazione su di essi, e li spedisce la modulo a seguire. In realtà in esso non transita nessun dato, vediamo il perché. Ogni modulo ha al suo interno un array di oggetti "Module"[7], sui quali invochera il metodo "inputCH(int[] input)", dove l'input passato è stato elaborato (in base alle funzionalità del modulo). In quest'ottica, quello che un "Pipe" deve effettuare è semplicemente l'inserimento nel modulo "mittente", del modulo "destinatario" dei dati (utilizzando i metodi definiti nella superclasse). Per far ciò sono stati ridefiniti alcuni metodi, in particolare "delete()" per la cancellazione della pipe.
Per quanto riguarda le proprietà impostabili su di esso, non è stata prevista alcun genere di impostazione.

### Derivate

L'idea alla base di questo modulo è molto semplice, ogni volta che si interagisce con il sensore tattile, in base al tipo di iterazione si ottiene una certa variazione di segnale (che misurata in volt può andare dai 10mv ad circa 0.7 v[8]). Quello che caratterizza un iterazione (tipo di pressione) con il sensore è la variazione di segnale prodotta, quindi la funzione matematica che mostra la misura di una variazione è proprio la derivata. Nel nostro caso la funzione che descrive il comportamento del segnale non è conosciuta, quindi non è possibile fare la sua derivata, però possiamo utilizzare il rapporto incrementale che è alla base del concetto di derivata:

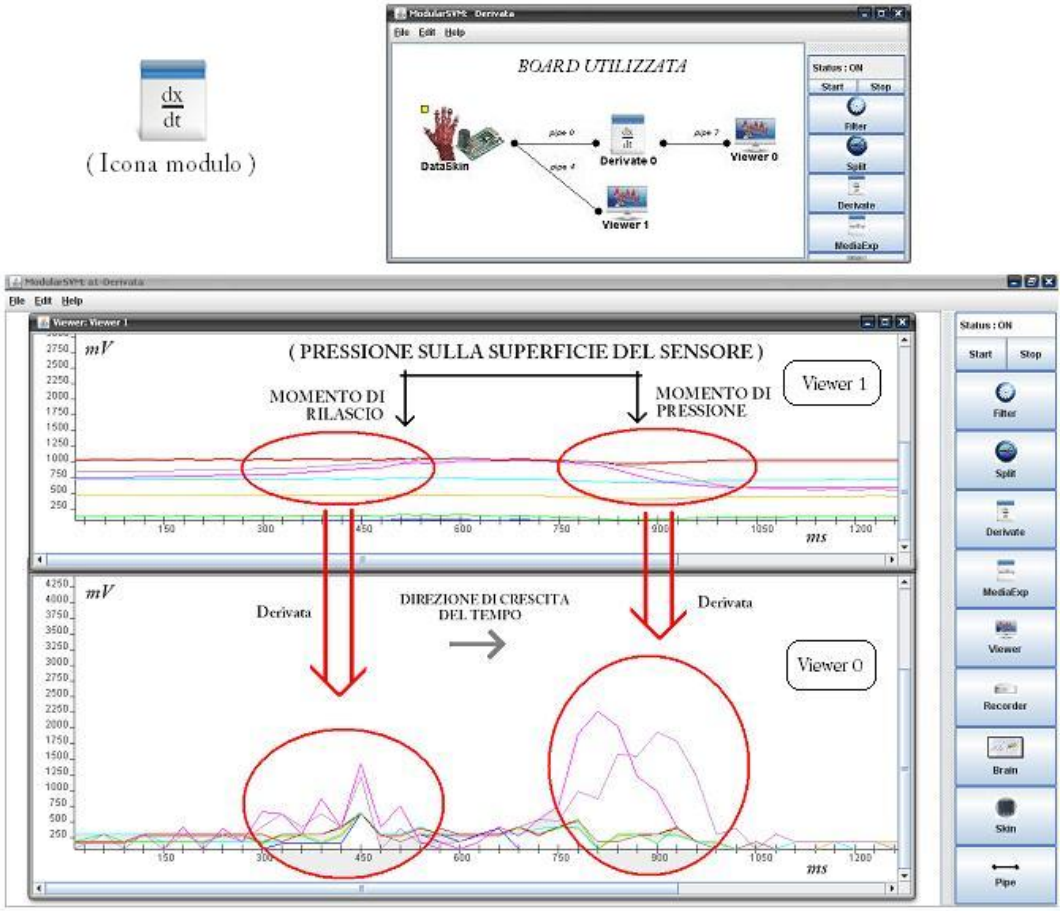
$$lim_{h\to 0}\,(x)\,\frac{f(x+h)-f(x)}{h}$$

Possiamo identificare $f\,(x)$ e $f\,(x+h)$ come due valori ricevuti in due instanti di tempo consecutivi, a distanza di tempo h (cioè l'inverso della frequenza di campionamento). In questo modo, il risultato ottenuto è la variazione del segnale nell'unità di tempo considerata. $\grave{E}$ possibile fare un'ulteriore considerazione, la frequenza di campionamento è costante nel tempo (utilizzo a regime), quindi anche il parametro $h$ è costante. Visto che $h$ è posto al numeratore, ed è uguale per ogni differenza fatta, è possibile porre $h = 1$ poiché ai fini di valutare la variazione tra i dati ricevuti, ciò non influisce. La funzione che viene calcolata da questo modulo è la seguente:

$$f\,(x+h) - f\,(x)$$

Quello che si ottiene è visibile nella seguente immagine:

[7]Vedi diagramma delle classi nella sezione: 5.3 a pagina 57
[8]Vedi sezione 2.3 a pagina 15




(Figura : Elaborazione dati del modulo “Derivate”)

In generale il classificatore prende dati da 0 a 5000 (convertiti in una scala da 0 a 1[9]), lavorare su un range piccolo di valori, può portare ad una difficoltà nel distinguere un dati utili dai disturbi introdotti dall'hardware. Per amplificare i dati ottenuti è stato introdotto nella proprietà del modulo la possibilità di impostare un coefficiente di amplificazione, in questo modo si può sfruttare meglio tutto il range a disposizione senza utilizzare un ulteriore modulo di amplificazione.

### Filter

Le funzionalità messe a disposizione da questo modulo sono le seguenti:

[9]Vedi sezione 4.5.2 a pagina 43

- zoom: amplifica il dato d'ingresso del fattore scelto (la scala va da 1 a 150);
- offset: introduce un offset nei dati di uscita al modulo;
- reduce: riduce i dati per in fattore impostato;
- tollerance: ha la stessa funzionalità di un filtro passa-alto, tutti i valori ricevuti uguali o minori del valore impostato vengono posti a zero. Questo parametro è molto utile per eliminare i disturbi presenti nel segnale.

L'applicazione di questi parametri avviene in contemporanea, ovvero prima viene valutato il parametro di "tollerance", dopo di che si moltiplica il dato per "zoom" e lo si divide per il valore di "reduce". Come ultima cosa viene sommato il valore di offset impostato.
In questo modo è possibile amplificare o ridurre i dati anche per valori non interi, questo modo di procedere è molto importante perché all'interno del software i dati ricevuti dal sensore sono di tipo "int", ciò comporterebbe una perdita di informazione qualrora si moltiplicasse per un valore non intero, mentre in questo modi c'è sempre perdita di informazione però l'errore commesso è minore.

**MedExp**

La funzione eseguita sui dati da questo modulo è la seguente:

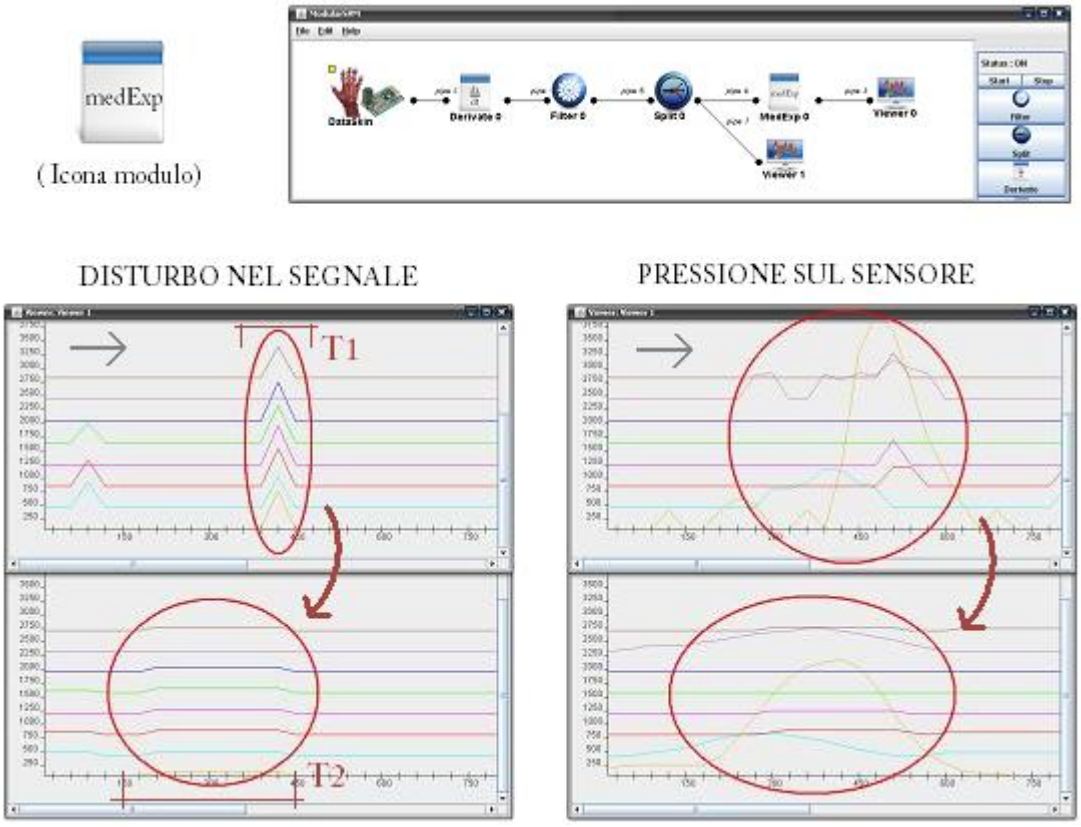
$$Out_n = X_n * p_n + X_{n-1} * p_{n-1} + X_{n-2} * p_{n-2} + ..... + X_{n-m} * p_{n-m}$$

$2 \leq m \leq 50$ (con una frequenza di campionamento di 20ms, impostando 50 come valore di m, si considera un intervallo di tempo di 1 secondo).

$p_n ... p_{n-m} = \frac{100}{m}$ (ovvero il peso del valore nella formula del valore, in questo caso dal valore più vecchio al più recente hanno o stesso peso, però in futuro è possibile modificare ciò per riconoscere particolari "forme" di input ).

dove:
**n**: rappresenta l'instante considerato (moltiplicato per la frequenza di campionamento di ottiene il tempo in cui i dato è stato ricevuto).
**m**: è la dimensione dei tempo presa in analisi (indica quanti valori vengono "ricordati" ed utilizzati per la media esponenziale).
Utilizzando questo modulo i picchi di valore sporadici hanno un impatto minore, mentre viene dato maggiore peso a valori minori ma continui nel tempo

(come ad esempio uno sfioramento sulla superficie del sensore). Il valore del parametro "m" può essere impostato dal menù proprietà del modulo. Nella figura che segue è illustrato un esempio del suo utilizzo (con $m = 8$):



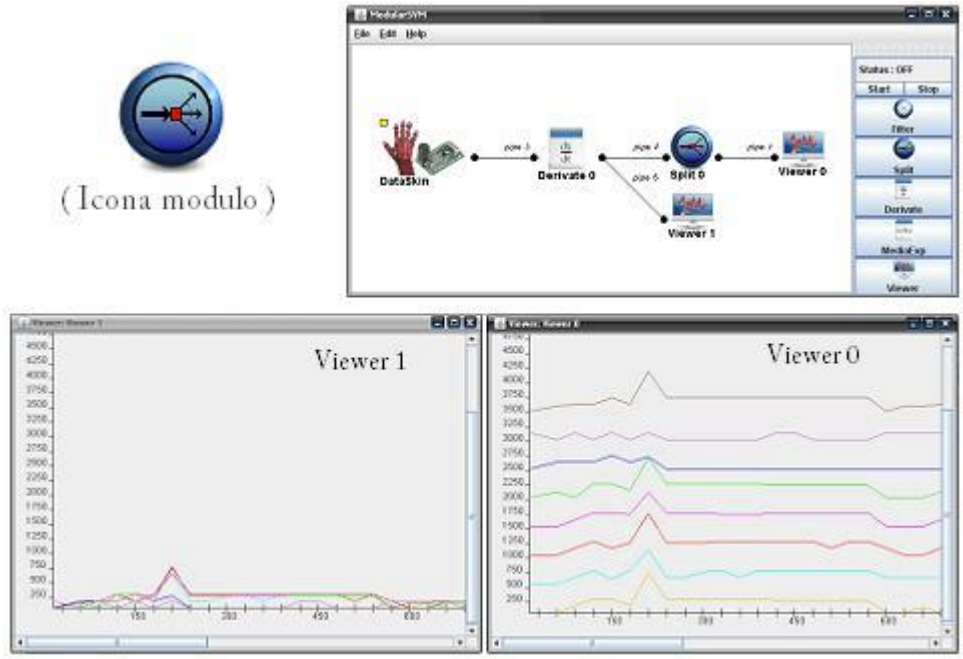

(Figura : Elaborazione dati del modulo "MedExp")

Come è stato appena detto, i picchi sporadici dovuti a disturbi del segnale vengono attenuati mentre, quando di esegue una pressione sul sensore essa risulta comunque evidente. L'effetto collaterale prodotto dal suo utilizzo è quello di creare una specie "d'eco dei valori", infatti è evidente che il tempo T1 (durata del disturbo) è breve, mentre il tempo T2 è molto più lungo (esattamente di "m" instanti). In conclusione, questa funzionalità è molto utile per evidenziare segnali continui nel tempo, e diminuire quelli sporadici come i disturbi. Occorre aver presente, che il suo utilizzo introduce un "delay" nei valori ottenuti, che potrebbe riperquotersi in una diminuzione dei tempi di risposta e in un prolungamento del segnale, ovvero quando si esegue una

pressione sul sensore, essa viene rilevata in ritardo, mentre quando la pressione sul sensore cessa, il software continua a rilevarla per un certo intervallo di tempo.

### Split

Non c'è migliore descrizione che quella visibile nella seguente figura:




(Figura : Elaborazione dati del modulo "Split")

In sostanza aggiunge ad ogni valore un offset, in modo tale da non essere sovrapposti, utile soprattutto dopo aver utilizzato il modulo "Derivate".

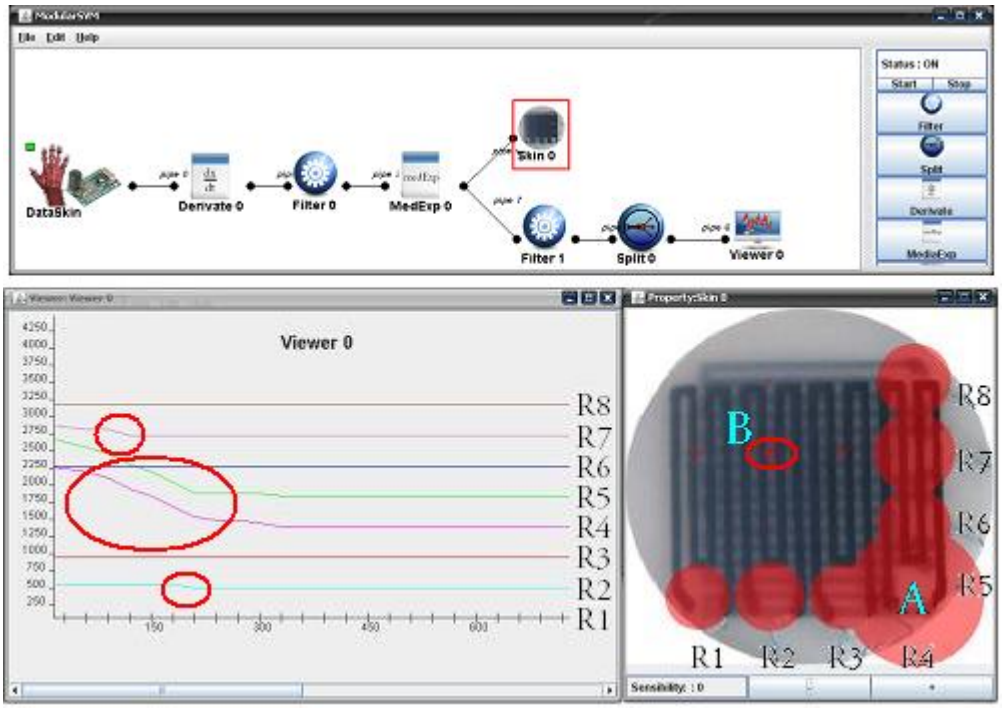
### Viewer

Mostra l'andamento dell'input nel tempo, dove sull'asse delle ascisse è riportata l'intervallo di arrivo del dato, mentre sull'asse delle ordinate il valore dei campi al suo interno. La parte grafica del module è contenuta nel packare "RealtimeViewer", in sostanza ogni qual volta si riceve un dato, tutti i dati precedentemente graficati subiscono uno "shift" a destra, ottenendo così l'effetto presente ad esempio, quando si visualizza l'utilizzo della CPU nel task manager. La dimensione la distanza temporale tra il primo è l'ultimo dato ricevuto è pari ad 1 secondo. Volendo si ha la possibilità di variarla, agendo

sul file “config.xml”, ma una dimensione eccessiva appesantisce la grafica del conponente diminuendo la “fluidità” di visualizzazione.

### Skin

Lo scopo di questo tool è visualizzare la zona dove avviene l’interazione con la pelle. Graficamente appare come:




(Figura : Aspetto grafico del modulo Skin)

Il sensore è visto come una matrice 4x4, dove il valore di ogni cella è dato dalla somma delle resistenze che si incrociano su di essa. Infatti è visibile la dimensione della zona A, cioè il punto di applicazione della pressione. Mentre la zona B, rappresenta un disturbo derivante l’interfaccia di collegamento con il sensore[10]. La dimensione della sfera disegnata è proporzionale al dato passato, non c’è alcun controllo sul valore massimo da passare, sarà compito dell’utilizzatore ridurre i dati in scala visualizzabile dal componente. Come per il modulo filtro è possibile impostare un valore minimo di tolleranza, sotto il quale non viene rappresentato nulla graficamente (cioè è messo a zero).

[10]Per ulteriori chiarimenti vedi sezione 2.2 a pagina 12

### Recorder

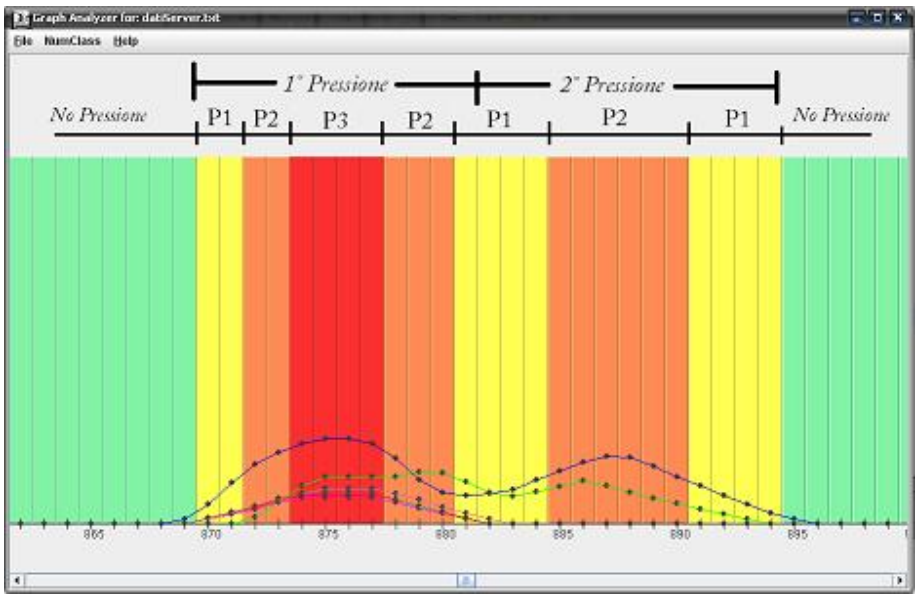
Questo modulo salva i dati ricevuti in ingresso in un file di testo, con nome corrispondente a quello impostato nel campo "DataFileServer" del file "config.xml". I dati vengono salvati in un particolare formato, in modo da essere utilizzati per le fasi successive di training e verifica del predict.

### Brain

Questo tool, analizza realtime i dati ricevuti, visualizzando in un finestra il risultato della classificazione. *É* possibile scegliere tra tutte configurazioni presenti nel file "config.xml" (dopo aver fatto il training), per dafault i ".model" devono risiedere nella directory impostata nel campo "pathDir" del file di configurazione. Inoltre, questo modulo invia i dati al componente successivo solo se il risultato della classifcazione è differente dalla "classe 1", cioè quando non c'è pressione sul sensore.

## 5.4 Tool: DataEditor

Questo tool permette di modificare le etichette dei dati memorizzati nel file "DatiServer.txt". L'aspetto grafico è il seguente:




(Figura : Interfaccia grafica dell'Editor)

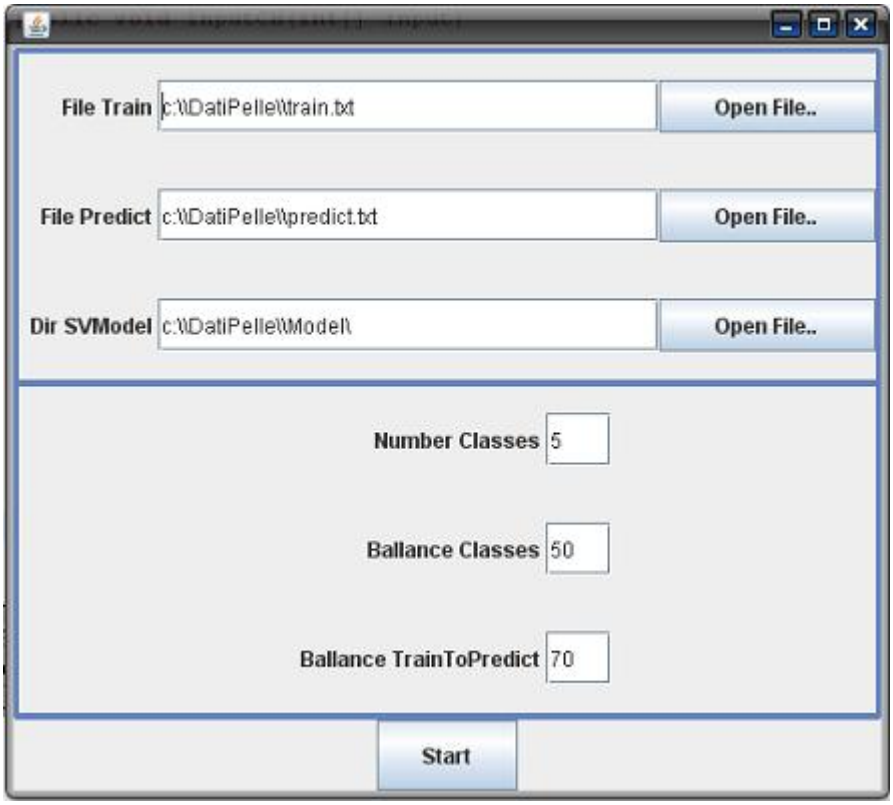
Ogni barra corrisponde ad una misurazione ricevuta, i punti presenti al suo interno sono i valori di ogni resistenza del sensore tattile, ed il colore della barra indica a quale classe appartiene. Per il momento è possibile distinguere tra 5 tipologie di classi, questo perché sono sufficienti ai fini del lavoro.
Nell'esempio in figura sono stati rappresentati due pressioni avvenute sullo stesso punto, una si seguito all'altra. Si può notare come la prima pressione sia di intensità maggiore rispetto alla seconda, infatti vengono coinvolte più tracce resistive del sensore, anche se il peso utilizzato è lo stesso. Questo perché la traccia resistiva impiega un certo tempo prima di tornare alla lunghezza originale. Alla seconda pressione si è avuto un allungamento inferiore del primo, anche se utilizzato lo stesso peso. L'interfaccia grafica offre le funzionalità di caricamento, salvataggio e impostazione del numero delle classi che in seguito il classificatore deve riconoscere. Di seguito viene riportata la relazione tra colore ed etichetta attribuita al dato:

- verde = 1 (nessuna iterazione col sensore)
- gialla = 2 (pressione lieve)
- arancione = 3 (pressione media)
- rossa = 4 (pressione massima)
- blue = 5 (sfioramento della superficie)

Per cambiare l'etichetta di un valore basta cliccare con il tasto sinistro (i colori sono cambiati ad ogni click), mentre per cancellare un dato si utilizza il tasto dentro. Il tasto centrale è stato previsto per impostare l'etichetta al solo volore 1 (per comodità di utilizzo).

## 5.5 Tool: TrainingSVM

Lo scopo di questo tool è effettuare in modo automatico la fase di training, per le diverse configurazioni presenti nel file "config.xml". Inoltre, testare sempre in modo automatico per ogni configurazione, la percentuale di errore commessa (fase di predict) e visualizzare il tutto in un report per facilitare la lettura dei dati così ottenuti. La prima interfaccia realizzata è la seguente:




(Figura : Interfaccia grafica dell'applicazione TrainingSVM)

La prima parte (in alto) serve per configurare i path da dove vengono letti i dati, e dove vengono salvati i risultati delle operazioni di trainig ed il report. Mentre nella seconda parte è possibile impostare il bilanciamento dei dati (da usare per il training e per la fase di predict), e la distribuzione dei dati per classi di appartenenza, per poter dare più peso ad una classe di dati rispetto alle altre. Di seguito sono descritte sinteticamente sia la fase di training che di predict, per fornire un'idea su cosa accade quando vengono eseguite.

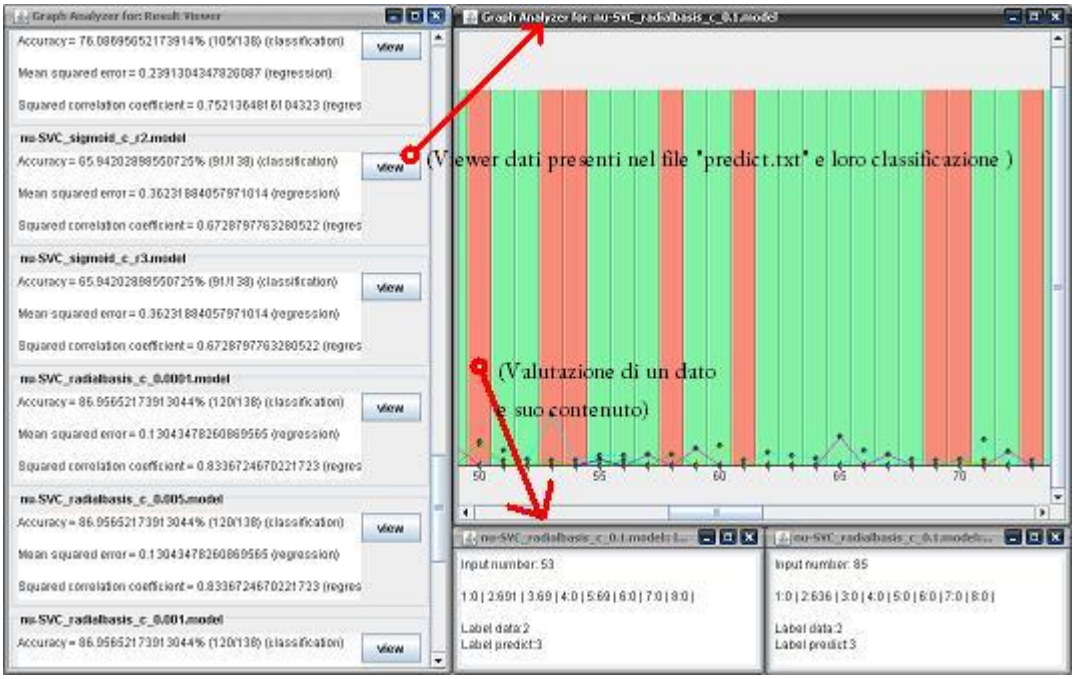
### 5.5.1 Fase di training

In questa fase, il software verifica quante differenti classi sono contenute nel file "DatiServer.txt", ed il numero di dati per ogni etichetta. Seleziona come numero massimo di elementi da prendere da ogni classe, questo numero massimo è scelto come il più piccolo valore del numero di dati (trovato in precedenza). Effettua una prima scelta in modo casuale dei dati per ogni

classe (ovviamente la classe con il numero minore di dati sarà inclusa totalmente). Dopo di che, in base ai parametri di bilanciamento impostati, vengono creati due file : “train.txt” e “predict.txt”. Il primo verrà utilizzato per l’addestramento delle SVM, i dati al suo interno sono a sua volta presi in modo randomico, i restati dati vengono utilizzati per la fase di predict. Alla fine dell’addestramento, per ogni configurazione viene generato il file “*.model” che rappresenta il modello ricavato dell’addestramento.

### 5.5.2 Fase di predict

In questa fase viene controllata la percentuale di errore nella classificazione, per fa ciò vengono utilizzati i restanti dei dati esculsi quelli usati nella fase di training. Dopo di che per ogni configurazione delle Support Vector Machines, viene creato un file contenente la lista delle classificazioni fatte. Ed in fine viene generato un report grafico per facilitare la visione dei risultati ottenuti. Il report generato dopo la fase di training e predict è cosi strutturato:




(Figura : Report generato dell’applicazione TrainingSVM)

*È* possibile visualizzare in un grafico i dati utilizzati nella fase di predict, la precentuale di successo delle diverse classificazioni fatte da ogni SVM (

impostata nel file “config.xml” ), identificando i dati che ha classificato in modo errato con una barra rossa, e l’etichetta erronea assegnata (cliccando con il mouse sulla barra). Questo grafico è uno strumento molto importante, perché in questo modo si ha la possibilità di capire se c’è una relazione tra tutti i dati non classificati correttamente. Ad esempio un problema riscontrato e risolto, si è avuto quando uno degli otto valori era molto alto (numericamente) ed i restanti molto vicini allo zero, il classificatore assegnava ad esso “poco peso” ed errava nella classificazione. In questo modo si ha la possibilità di correggere questi errori, ad esempio svolgendo di nuovo la fase di addestramento dopo aver modificato tali valori, ma anche aggiungendo valori nuovi (modificando il file di training).

# Capitolo 6

# Test e validazione del sistema

L'obiettivo del test è costruire e validare un classificatore che sia in grado di discriminare quattro tipi di interazioni con la pelle, che sono: *pressione lieve*, *pressione normale*, *forte pressione*, *sfioramento*. In realtà le tipologie di interazioni con la pelle sono due, ma è interessante riuscire a capire se è possibile fare delle "sfumature" in queste tipologie di interazioni. *´E* molto importante capire il concetto di "classe di appartenenza", esso è alla base per costruire un buon classificatore. Diciamo che ogni classe ha una particolare caratteristica che la distingue dalle altre classi, obiettivo dei test è in primo luogo identificare questa caratteristica per ogni classe, dopo di che creare una board[1] attraverso la quale ogni dato ricevuto viene filtrato, diminuendo i rumori presenti ed esaltando ciò che caratterizza le quattro classi. In questo modo risulterà più evidente la classe di appartenenza per ogni dato ricevuto dal sensore, semplificando la vita del classificatore SVM.

## 6.1 Descrizione dell'ambiente di test

I test sono stati eseguiti utilizzando dei pesi per simulare le pressioni, mentre per quanto riguarda "la carezza del sensore" si sono adottate due strategie: la prima è quella di far scivolare il peso sulla superficie del sensore, mentre la seconda corrisponde proprio ad eseguire manualmente lo sfioramento sul sensore. Questo perché non essendo dotati di nessuno strumento che esegua lo sfioramento in modo automatico, agire manualmente risulta una buon campo di test. La strumentazione utilizzata è composta da un set di pesi che valori: 10gr,15gr,20gr,45gr. Il peso massimo testato è 60gr, mentre il

[1]Col termine board si intende una configurazione di moduli appartenenti al tool ModularWorkBoard col fine di fare preprocessing dei dati ricevuti, vedi 5.3 a pagina 55

minimo è di 15gr. Da notare che i pesi non sono perfettamente tutti uguali, ma da una pesa di essi si è visto che possono variare il loro valore nominale di circa $\pm 2gr$. Questo però non influisce sui risultati ottenuti perché una tale differenza è poco apprezzabile dal prototipo di sensore utilizzato.

Per quanto riguarda l'esecuzione dei test, il valore da considerare non è il peso posto sul sensore ma la pressione esercitata sulla superficie dello stesso. Questo perché in linea di principio, il tipo di risposta dal sensore è data dalla pressione effettuata su di esso e dal numero di resistenza che si vanno a sollecitare.

Una volta fissata la board, per ottenere il classificatore si sono seguiti i seguenti passi:

1. Ogni peso viene fatto cadere da un altezza di circa 2 cm dal sensore, tramite l'utilizzo di un piccolo trepiedi.

2. Le misurazioni sono fatte solo sulle resistenze più superficiali (R1,R2,R3 e R4)[2], ripetendo i lanci su zone il più possibile casuali.

3. Individuare i valori di pesi per cui si raggiunge il fondoscala dei valori del sensore, ed il peso che rappresenta la sensibilità minima.

4. Per ogni resistenza sono effettuati 20 "lanci di peso".

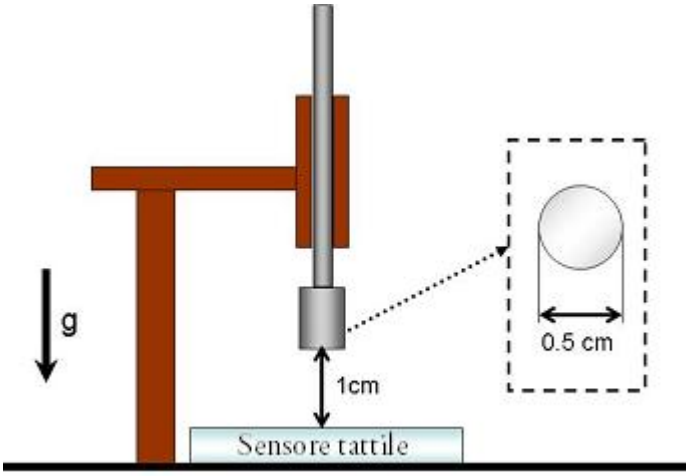
5. Al termine di ciò, viene utilizzato l'editor per selezionare le etichette sui dati raccolti, e contando quante pressioni sono identificabili. In questo modo, si ha un riscontro immediato della sensibilità delle resistenze (cioè su 20 lanci quanti sono visibili).

6. A questo punto si generano i file per la fase di apprendimento e per la fase di verifica.

7. Si seleziona la migliore configurazione di SVM e si procede con un test "dal vivo", registrando il numero di classificazioni giuste ed errate, e la loro zona sul sensore.

8. Una volta testate tutte le classi singolarmente (prima ogni peso scelto per rappresentare la classe), viene fatta una "merge" di tutti i file utilizzati per la fase di trainig e predict e si verifica la precisione del classificatore così ottenuto. In questo capitolo sono riportati solo i dati migliori ottenuti dopo diverse prove.

[2]Vedi sezione 2.1 a pagina 11




(Figura 6.1: Schema della strumentazione utilizzata per il test)

Per quanto riguarda le prove dello "sfioramento", esse saranno eseguite prima lungo le direzioni delle resistenze (sia orizzontali che verticali), poi con traiettorie casuali sul sensore cercando di coprire tutte le zone. Verra diversificata anche il tipo di sfioramento, cioè una sola "passata"sul sensore, due "passate" oppure a passata multipla sempre sulla stessa zona oppure su una zona differente.
L'ultima serie di test sarà quella di adagiare il sensore su una superficie non piatta (situazione di utilizzo dei test precedenti), ad esempio adattandolo su superfici curve oppure spigolose, come un cilindro o il bordo di un tavolo. Questo serve per verificare una caratteristica molto importante del sensore, cioè l'adattabilità alle superfici di utilizzo (questo uno dei motivi per cui è stato scelto di realizzarlo in materiale siliconide ). In questo modo cercheremo di coprire tutti i principali modi di utilizzo, al fine di capire quali sono le potenzialità della pelle e del classificatore.
Inoltre è molto importante ricordare che la sensibilità di questo prototipo di sensore non è omogenea su tutta la superficie, questo perché le resistenze più esterne sono più sensibili rispetto a quelle più interne, situazione descritta nel capitolo 2.3 a pagina 15. Un altro motivo è dato dall'usura derivante l'utilizzo, il prototipo usato per i test è gia stato usato nel lavoro di testi relativo ai sensori skin-like[3], ed ha subito uno stress soprattutto in alcune zone, queste zone sono le meno sensibili del sensore, per cui i dati ricavati dall'interazione con queste zone vanno contestualizzati alla sensibilità della resistenza. In questo test non è stato testato solo il sensore ma anche come i

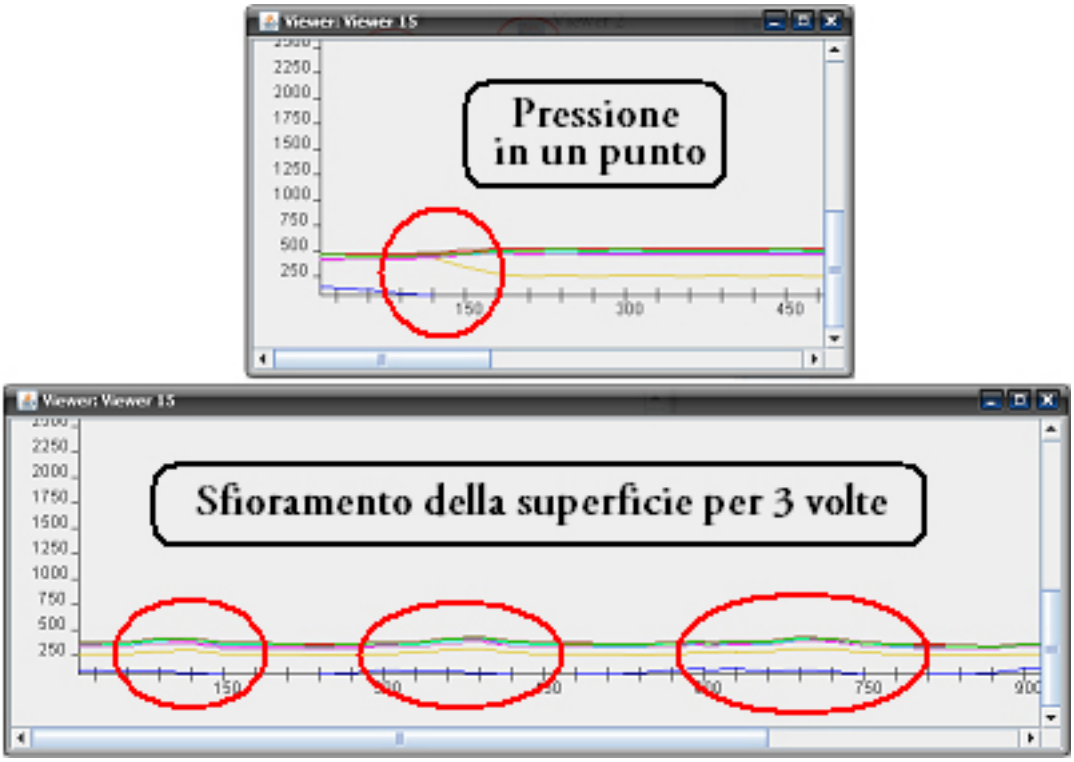
[3]Vedi sezione 2.3 a pagina 15

tool sviluppati, permettendo di corregge alcuni errori presenti nel software. Per quanto riguarda la validazione dei test, quello che si è fatto trovare una configurazione di pesi e tempo tra due pressioni, per cui l'esperimento ripetuto aveva lo stesso risultato. *É* stato notato che per pesi piccoli, il minimo tempo di attesa tra due pressioni è 30 secondi. Mentre per pesi grandi, è stato richiesta un attesa di almeno 1 minuto. Ricordiamo che questo fenomeno è dovuto principalmente all'isteresi del sensore[4].

## 6.2 Creazione board

La creazione della board è una delle fasi più importante per realizzare un buon classificatore. L'obiettivo finale del data flow creato, è quello di avere un un segnale costante quando il sensore non subisce pressione di nessun genere, ed una variazione dei valori proporzionale alla pressione effettuata su di esso. Il componente fondamentale per questo genere di analisi è "Derivate", attraverso esso è possibile visualizzare la variazione del segnale. Da quanto detto nel capitolo 2, sull'interfaccia di collegamento del sensore, il segnale campionato è affetto da rumore, il quale è stato buona parte eliminato dal campionamento a 50Hz. I segnale ricevuto dal sensore, in generale non è ma perfettamente stabile, ma dente a subire variazioni dovute in parte all'interfaccia del sensore, in parte al sensore stesso. Questo secondo comportamento è molto evidente quando, dopo un periodo di inutilizzo del sensore, lo stesso avrà la stessa temperatura dell'ambiente circostante. Quando si collega l'alimetazione all'interfaccia, la stessa alimenta le resistenze all'interno del sensore, le quali producono calore. Visto che la variazione di temperatura delle piste resistive comporta una variazione del valore delle resistenze, quello che si ottiene è una variazione del segnale molto simile a quelle generata da una pressione sulla sua superficie. LA durata di questo periodo di equilibrio instabile, dipende dalla temperatura di partenza del sensore. *É* stato riscontrato che il tempo medio per stabilizzare il segnale, quando il sensore è in riposo in un ambiente a temperatura di circa 25° è di circa 20 minuti. Tale tempo aumento se la temperatura di partenza è più bassa (inferiore ai 20°), mentre rimane costante per temperature fino a 35° (per temperature maggiori non è stata fatta alcuna prova).
Andiamo subito al sodo, di seguito sono riportate le screenshot delle due tipologie di segnali da classificare:

[4]Vedi vedi 2.1 a pagina 10




(Figura 6.2: Nel primo grafico è rappresentato l'output del sensore dopo da una pressione su un punto sulla superficie. Il secondo grafico rappresenta l'output del sensore dopo aver subito uno sfioramento della superficie per 3 volte consecutive.)

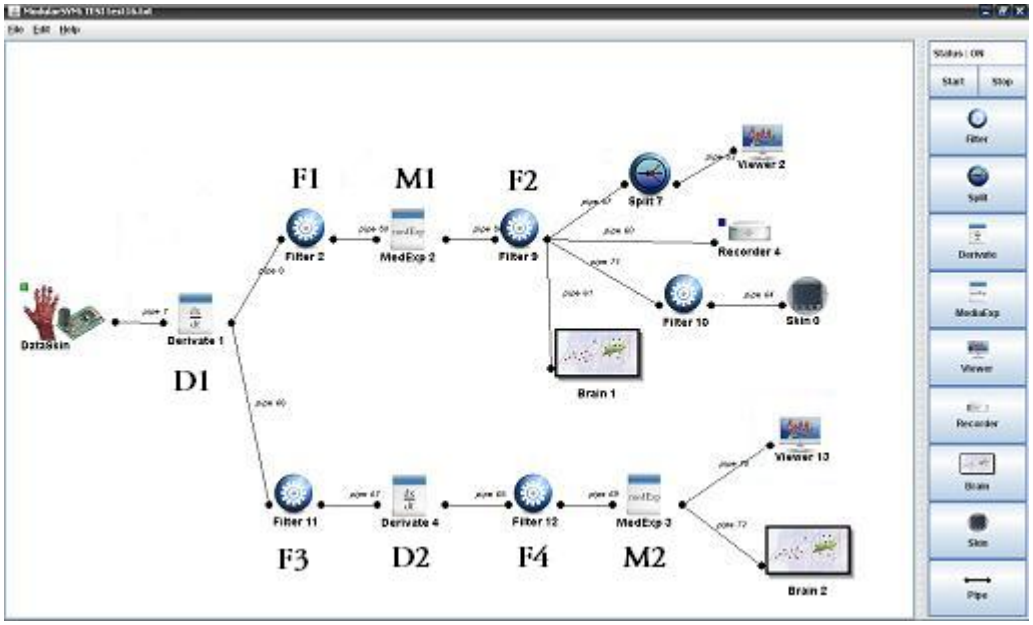
La differenza principale tra i due modi di interazione con la pelle, consiste nel numero di resistenze che sono interessate dalla pressione. Infatti nel primo grafico della figura 6.2, le resistenze interessate sono due (linea gialla e linea blue), le altre hanno una piccola variazione. Mentre nel secondo grafico della figura 6.2, quasi tutte le resistenze hanno una variazione simile, ripetuta per tre volte. Ricordiamo che il sensore ha un tempo di isteresi, entro il quali la resistenza, dopo aver subito la pressione e di conseguenza un allungamento $\Delta l$, impiega un certo tempo per tornare alla lunghezza originale. Se si eseguono due pressioni consecutive, quello che accade è che la prima sarà ben evidente, mentre la secondo sarà appena accennata. Questo perché la resistenza può allungarsi si un $\Delta l_{max}$, una volta raggiunto anche se si continua ad esercitare pressione (sempre sullo stesso punto) non c'è più variazione di lunghezza della resistenza, e di conseguenza l'output del sensore

rimane invariato. Questo fenomeno non è presente invece con lo sfioramento, perché accarezzando tutta una superficie si provoca una variazione più o meno intensa sulla zona, ma tale variazione non è consistente da raggiungere $\Delta l_{max}$ su tutte le zone interessate (con una pressione di intensità media), quindi è possibile effettuare una sequenza molto lunga di sfioramenti prima che il sensore inizia a perdere sensibilità.
L'andamento del segnale ricevuto dal sensore dopo una pressione in un punto è molto simile a quello prodotto dallo sfioramento della superficie. L'uniche differenze sono una l'ampiezza della variazione (maggiore quando si esercita una forte pressione), e l'altra è durata di tale variazione, che nel caso dello sfioramento può essere molto lungo rispetto alla pressione. Questo ci fa intuire che il classificatore difficilmente riuscirà a distinguere una pressione da uno sfioramento, solo valutando l'ampiezza della variazione, ma occorre valutare anche il tempo. Il modulo che si occupa di ciò, è quello che effettua la media esponenziale dei dati. In base al numero di instanti di tempo considerati è possibile evidenziare maggiormente una variazione duratura, più tosto che una variazione breve ma più intensa. Inoltre, durante lo sfioramento si provocano variazione piccole, ma valore spesso differenti e molto frequenti, quindi ciò suggerisce un analisi del segnale con alla seconda derivata (per quanto riguarda lo sfioramento).
A questo punto, è stata realizzata una prima "board", dove con un unico flusso si è realizzato un classificatore che distinguesse i tre livelli di pressione e lo sfioramento. I risultati ottenuti erano appena sufficienti, per due motivi:

- la percentuale di errore sulla classificazione era alta, quando si trattava di distinguere uno sfioramento da uno pressione di media intensità (utilizzo di pesi da 20g a 30g), circa il 50%.

- il tempo di risposta del sistema era anch'esso alto, questo dovuto all'introduzione della media esponenziale con un valori di 20 instanti di tempo (con frequenza di campionamento a 20ms, si introduceva un ritardo nella risposta di circa 0.4 secondi)

Per questi motivi, è stato scelto di realizzare una seconda board, dove sono presenti due flussi di dati, uno specializzato per rilevare i diversi gradi di pressione, mentre l'altro specializzato nel rilevare lo sfioramento della superficie. Di seguito è riportata la board realizzata:

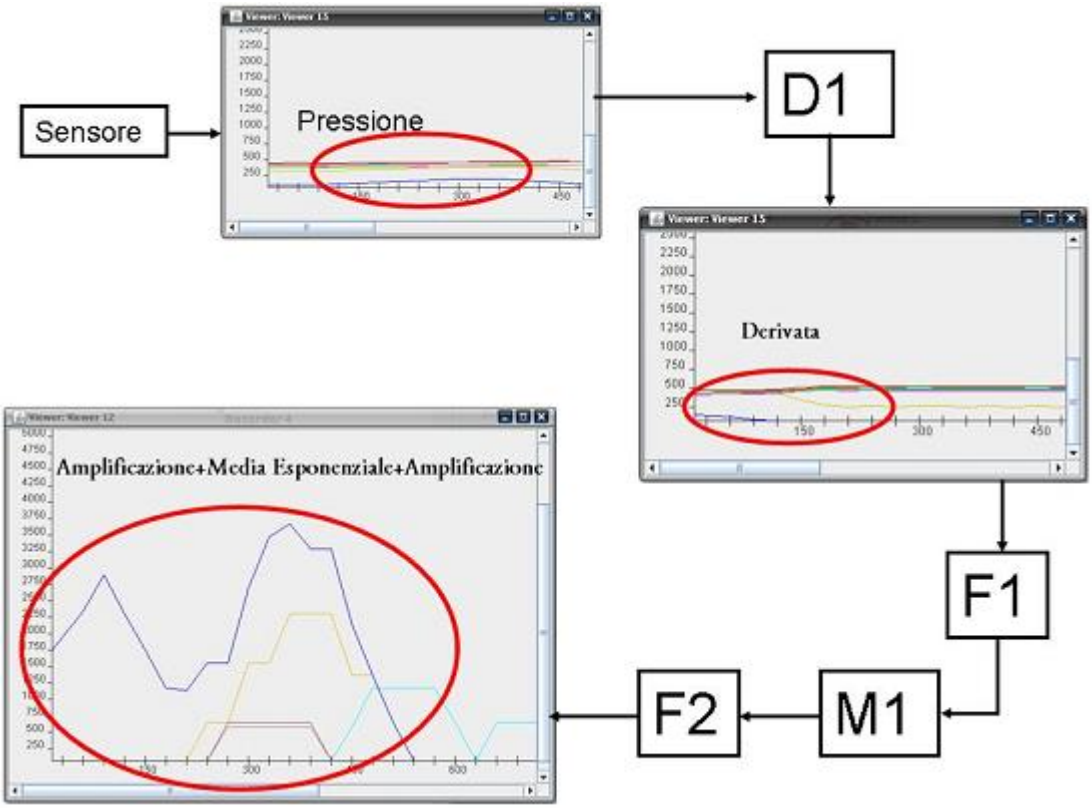



(Figura 6.3: Board realizzata per la classificazione della pressione (flusso in alto), per la classificazione dello sfioramento (flusso in basso) )

Di seguito sono riportate le impostazioni dei principali moduli presenti nella board:

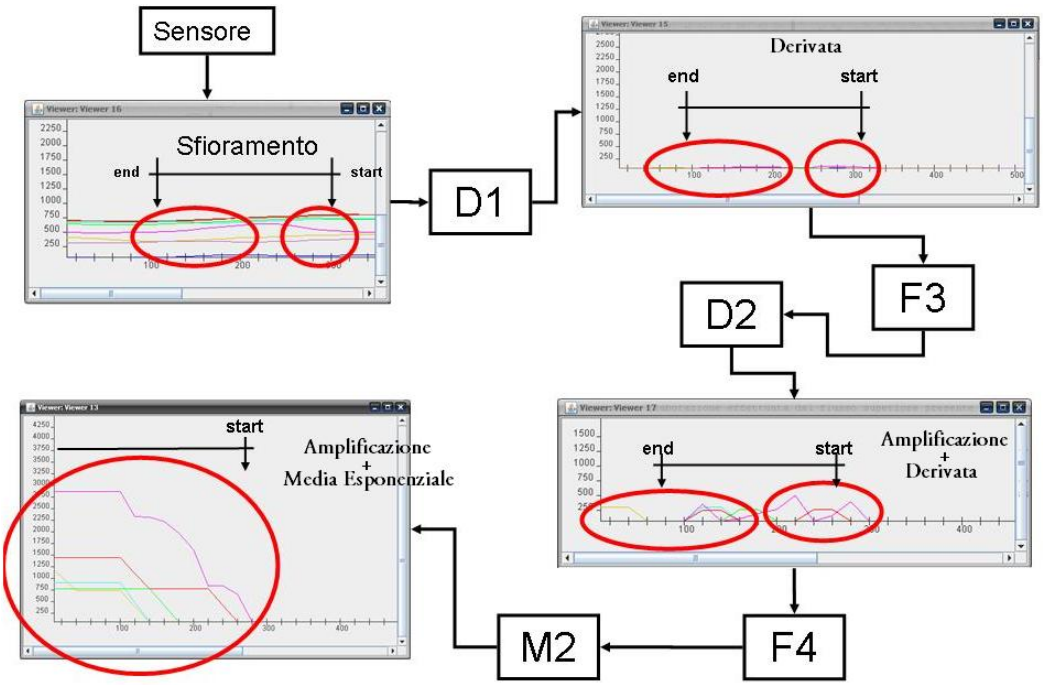
- F1: Zoom: 10 Reduce: 1 Offset:0 Tollerance: 10
- F2: Zoom: 14 Reduce: 1 Offset:0 Tollerance: 0
- F3: Zoom: 10 Reduce: 1 Offset:0 Tollerance: 15
- F4: Zoom: 15 Reduce: 1 Offset:0 Tollerance: 0
- M1: CoeffTemp: 5
- M2: CoeffTemp: 20

Nelle figura che seguono sono riportati i diversi output generati dai moduli, per quanto riguardo sia la pressione che lo sfioramento.




(Figura 6.4: Elaborazione effettuata dal flusso superiore presente nella board)

La sequenza di moduli, amplifica la variazione dovuta alla pressione sul incrocio delle resistenza R1-R5. Mentre i segnali relativi alle altre piste resistive, anche se hanno subito una piccola variazione non risultano evidenti come le resistenze interessate dalla pressione (escluso R2 ed R6 che sono molto vicine alla zona di pressione).




(Figura 6.5: Elaborazione effettuata dal flusso inferiore presente nella board)

in questo caso, è evidente il ritardo introdotto dal modulo che esegue la media esponenziale. Infatti la durata dell'input è stata prolungata di 20 instanti di tempo. L'effetto positivo, è l'amplificazione prodotta sui dati continui nel tempo per almeno il valore impostato nella media esponenziale. Tale ritardo è ammissibile per identificare lo sfioramento, questo perché la durata di uno sfioramento può essere molto più lunga del ritardo introdotto, quindi viene avvertito meno (soprattutto se si sfiora il sensore più volte).

Prima di procedere con la fase di test vera e propria, occorre ripassare l'indispensabile di fisica, per poter valutare la pressione esercitata sul sensore. La pressione su un area è data da:

$$P = \frac{F}{A} \qquad (5.1)$$

dove $F$ è la forza misurata in Newton, mentre $A$ è l'area sulla quale viene applicata. *É* possibile ricavare la forza dalla legge che descrive il *Lavoro*:

$$L = h \times F \qquad \text{da cui: } F = \frac{L}{h} \qquad (5.2)$$

dove $h$ è lo spostamento prodotto dalla forza sul corpo. Inoltre sappiamo che il *Lavoro* di un corpo in caduta libero, può essere espresso anche come:

$$L = m \times *g \times *h \qquad \text{che sostituendo nella 5.2 si ottiene:}$$
$$F = \frac{m \times g \times h}{h} = m \times g \quad (5.3)$$

Da cui la 5.1 si ha:

$$P = \frac{F}{A} = \frac{m \times g}{A}(N/m^2) \quad (5.4)$$

In tutti i test, i pesi utilizzati hanno in fonda una "*testa*" a forma di cerchio con raggio fisso di 0.25cm. Quindi nel nostro caso l'area vale: $A = r^2\Pi = 0.196cm^2 \Rightarrow 1.96 \times 10^{-5}m^2$.

## 6.3 Test : pressione

La prima di tutto, sono stati individuati i pesi per cui si ha sensibilità minima (15 gr), e sensibilità massima (45 gr). Per valori inferiori al minimo, il sensore rileva la pressione solo sulle resistenze più sensibili, mentre per le altre non si avverte nessuna variazione. In realtà analizzando il segnale direttamente dal sensore (senza processing dei dati), anche con pesi di valore di 10gr è possibile individuare variazioni nel segnale, ma tali variazioni hanno la stessa entità di quelle prodotto dai disturbi del segnale, quindi vengono annullate durante la fase di processing. Per quanto riguarda la scelta del peso intermedio, effettuando prove sperimentali è stato individuato il valore di 25 gr. Quindi ricapitolando, le pressioni con le quali si effetuano i test sono:

- 15 gr (pressione minima $\Rightarrow 75.05 \times 10^{-2}N/m^2$ (Classe 1))
- 25 gr (pressione media $\Rightarrow 125.12 \times 10^{-2}N/m^2$ (Classe 2))
- 45 gr (pressione massima $\Rightarrow 225.22 \times 10^{-2}N/m^2$ (Classe 3))

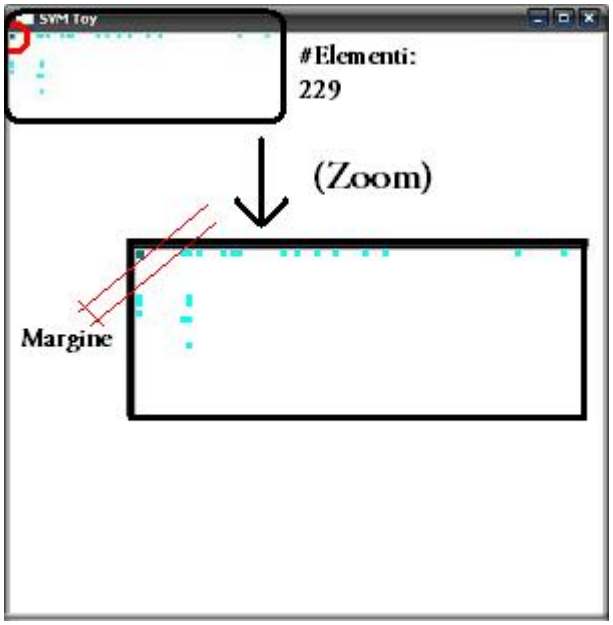
Per ottenere il training set sulle tre classi di pressioni (così identificate), sono stati prima raccolti i dati *separatamente* per le tre classi, per poi essere etichettati con l'utilizzo dell'editor. Infine, tali training set, sono stati unificati in un unico file, ed eseguito l'addestramento con il nuovo training set, così ottenuto.

Di seguito sono riportati i dati di training ottenuti per le tra classi, e la classificazione fatta su di essi. Per la visualizzazione è stato utilizzato uno degli strumenti inclusi nella libreria LIBSVM, chiamato "SVMToy". Tale tool, legge i training set e lo visualizza sullo schermo un punto per ogni

elemento del training set, dove in base alla classe di appartenenza viene attribuito un colore. La funzionalità più interessante di questo tool, è l'esecuzione di una configurazione di svm visualizzando come sono classifica i dati attraverso l'utilizzo di fasce colorate.

### 6.3.1 Pressione 15 gr, Classe 1

Sono state eseguite un numero di 40 pressioni sulla superficie, divise equamente sulle resistenze R1,R2,R3 e R4. La distanza che intercorre tra una pressioni consecutive è di 30 secondi. I punti di applicazione delle pressioni hanno coperto tutta la lunghezza della resistenza, evidenziando zone piu sensibili rispetto ad altre. I risultati ottenuti sono i seguenti:

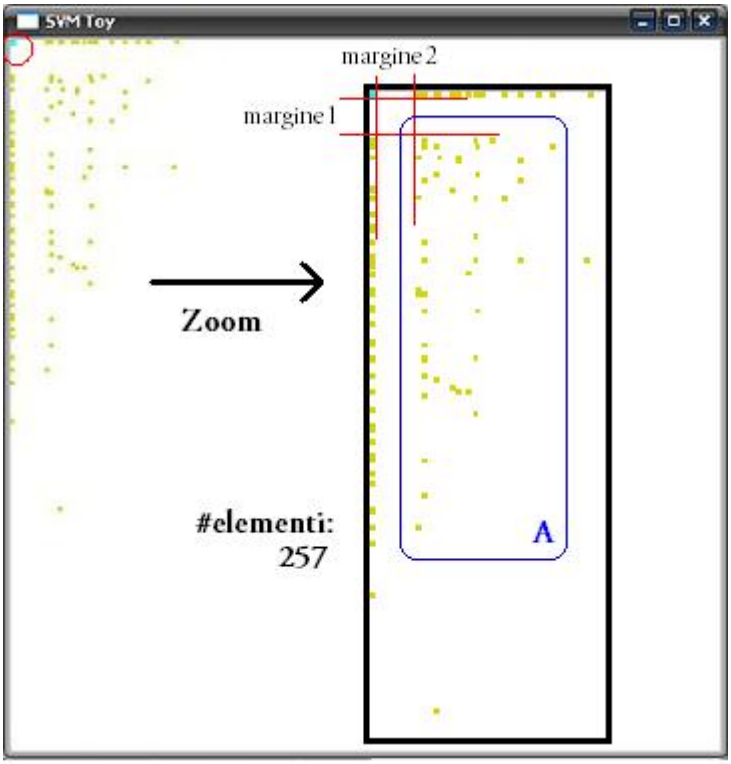



(Figura 6.6: Training set per la classe 1 (pressione minima) )

Nella figura salta subito all'occhio che la "*classe 0*" (cioè nessuna pressione sul sensore), o visualizzata in un sono punto (in altro a destra), questo perché il processing fatto dalla board, elimina i disturbi del segnale, quindi i valori numerici sono tutti zero. Inoltre è presente margine di separazione tra le due classi, questo fa intuire che la classificazione per questo caso sarà molto buona. La dimensione della classe 1 è di 229 elementi, nell'immagini sembra molto meno, questo perché molti valori sono sovrapposti. Questo è dovuto al processing dei dati, che filtrando i valori per poi amplificarli, porta ad avere dei valori ben precisi, eliminando i valori intermedi.

### 6.3.2 Pressione 25 gr, Classe 2

Anche in questo caso sono state eseguite 40 pressioni spare uniformemente su tutta la superficie del sensore. Il training set generato è il seguente:

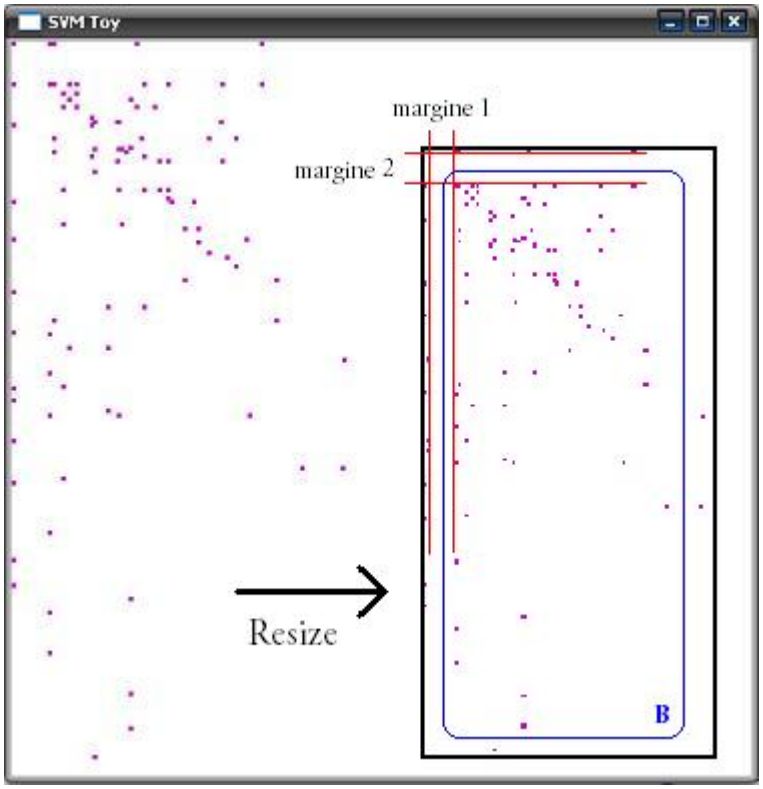



(Figura 6.7: Training set per la classe 2 (pressione media) )

In questo caso il numero di elementi ottenuti, è maggiore rispetto a quelli presenti nel training set della classe 1. Questo perché, con questa i dati generati da una pressione di intensità maggiore, sono in numero maggiori. Dall'immagine del training set possiamo notare la presenza di di due margini, uno orizzontale e l'altro verticale. Nel training set c'è una separazione tra la classe 2 e la classe 1 (come ne caso precedente), in particolare i dati sono concentrati nella zona A, e sulle zone laterali. Questa distribuzione fa intendere che non tutte le resistenze hanno risposto con la stessa intensità, in particolare è stata riscontrata una bassa sensibilità da parte delle resistenze R3, ed R7.

### 6.3.3 Pressione 45 gr, Classe 3

Come per i test precedenti, sono state eseguito 40 pressioni distribuite uniformemente sulla superficie:

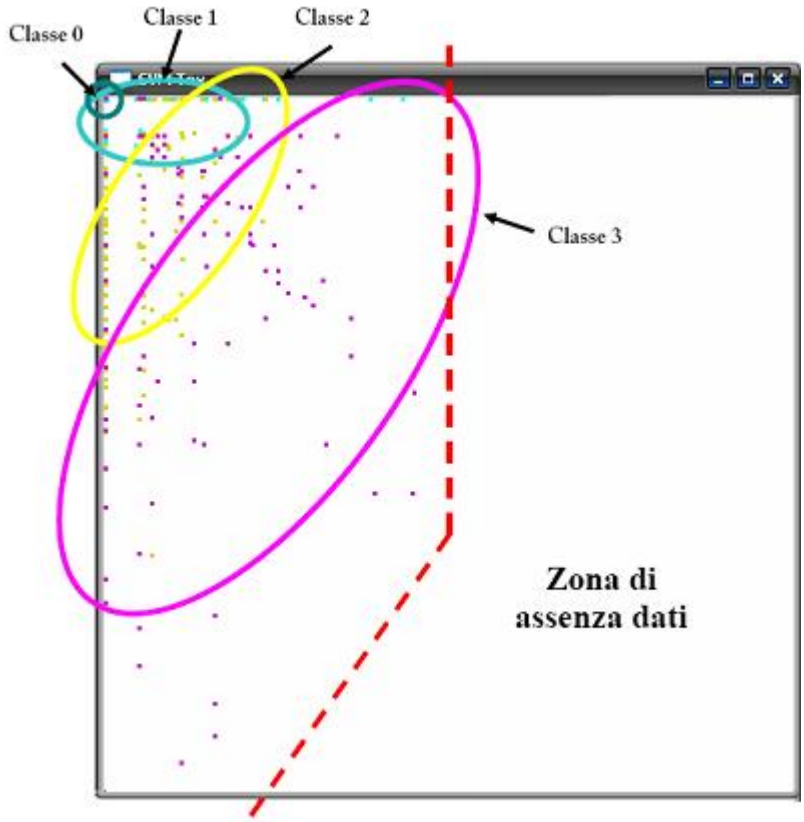



(Figura 6.8: Training set per la classe 3 (pressione massima) )

Anche questo training set ha due margini (orizzontale e verticale), i quali risultano di dimensione simile ad i margini riscontrati nei casi precedenti. La differenza tra i dati del training set attuale e quello della classe 2, è la dimensione dell'area. Infatti si nota molto bene che la "*zona B*" risulta maggiore rispetto alla "zona A" della classe 2.
Il numero di elementi del training è minore rispetto agli casi perché, durante la fase di editor dei dati, i dati relativi alle classi intermedie vengono eliminati, lasciando solo i valori più alti. In questo modo è l'addestramento delle SVM risulta più pulito, il che permetterà una maggiore precisione nella classificazione.

### 6.3.4 Addestramento e test

Vediamo il training set generato dall'unione dei "set" precedenti:




(Figura 6.9: Training set per la classificazione dei livelli di pressione )

Gli elementi delle classi sono parzialmente sovrapposti, quindi non sono perfettamente distinti. Ciò che è interessante invece, e vedere che buona parte della meta destra del quadrante è vuota, nessun valore ricade al suo interno. Questo in genere è dovuto al fatto che una pressione in un punto in generale coinvolge solo poche resistenze, le altre hanno valori nulli, oppure molto inferiori a quelli delle resistenze interessate da ciò.
Di seguito sono riportati la lista di configurazioni con le quali viene eseguito l'addestramento.

1. `nu-SVC_polynomial_d1.model`
2. `nu-SVC_linear.model`
3. `nu-SVC_polynomial_d1.model`

4. nu-SVC_polynomial_d2.model
5. nu-SVC_polynomial_d3.model
6. nu-SVC_polynomial_d4.model
7. nu-SVC_polynomial_d5.model
8. nu-SVC_polynomial_d6.model
9. nu-SVC_polynomial_d7.model
10. nu-SVC_polynomial_d8.model
11. nu-SVC_polynomial_d2_g5.1.model
12. nu-SVC_polynomial_d2_g5.1_n0.5_e0.11.model
13. nu-SVC_radialbasis_g0.5.model
14. nu-SVC_radialbasis.model
15. nu-SVC_radialbasis_c_0.0001.model
16. nu-SVC_radialbasis_c_0.005.model
17. nu-SVC_radialbasis_c_0.001.model
18. nu-SVC_radialbasis_c_5.79.model
19. nu-SVC_radialbasis_c_0.1.model
20. nu-SVC_radialbasis_c_1.model
21. nu-SVC_radialbasis_c_10.model
22. nu-SVC_radialbasis_c_100.model
23. nu-SVC_radialbasis_c_1_g_10.model
24. nu-SVC_radialbasis_c_5_g_10.model
25. nu-SVC_radialbasis_c_100.model
26. nu-SVC_sigmoid_c_r0.model
27. nu-SVC_sigmoid_c_r1.model
28. nu-SVC_sigmoid_c_r2.model
29. nu-SVC_sigmoid_c_r3.model
30. nu-SVC_sigmoid_c_100_g_0.01.model
31. nu-SVC_sigmoid_c_50_g_0.01.model
32. nu-SVC_sigmoid_c_40_g_0.01.model
33. nu-SVC_sigmoid_c_500_g_0.01.model

A questi punto si esegue il tool “TrainingSVM”, per la creazione dei file di training e predict, ed eseguendo in modo automatico le diverse configurazioni di svm. I risultati ottenuti sono i seguetenti:

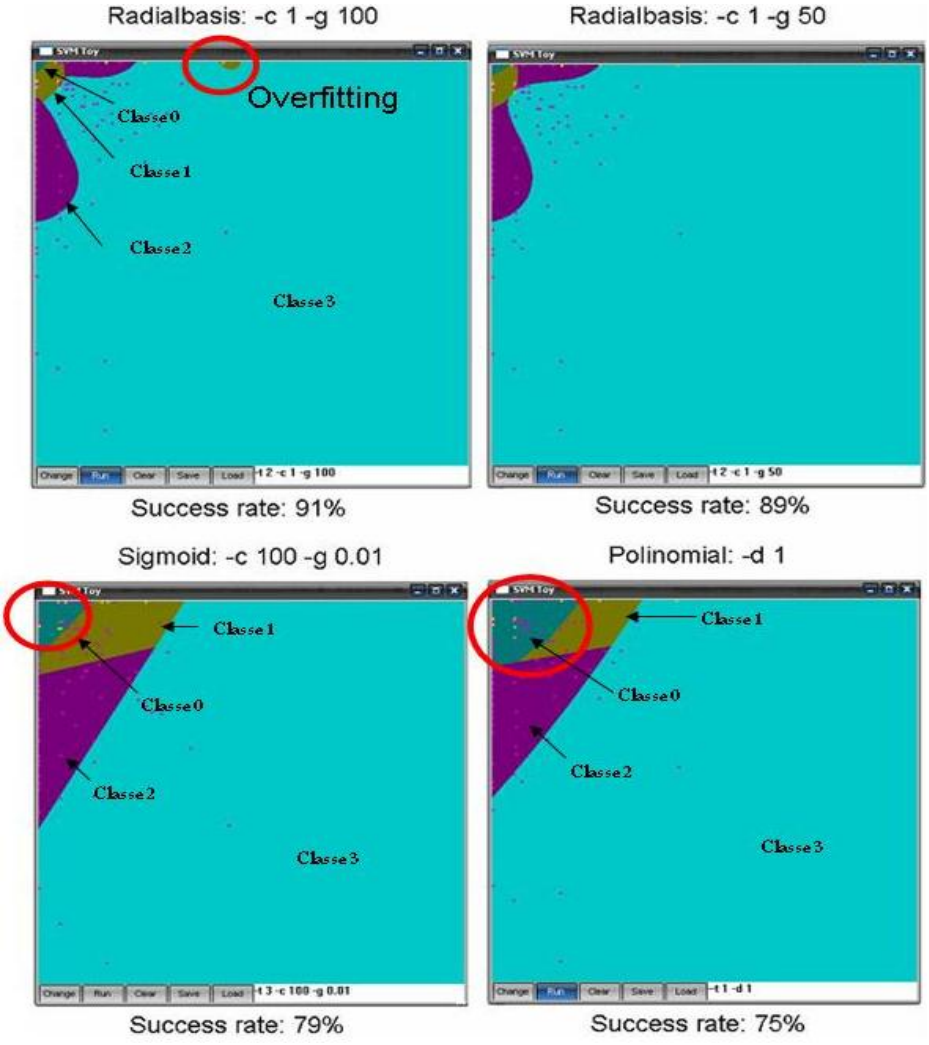
**train.txt** Questo file insieme contiene al suo interno 484 elementi, di cui distribuiti equamente fra tutte le classi (121 elementi per classe)

**predict.txt** Contiene 248 elementi, divisi anch'essi equamente (62 elementi per classe)

**Risultati dell'apprendimento** Le configurazioni migliori per i diversi kernel sono state le seguenti:

- "radialbasis": "*-t 2 -c 1 -g 100*" con successo del 91%, e "*-t 2 -c 1 -g 50*" con successo del 89%.
- "sigmoid" : "*-t 3 -c 100 -g 0.01*" con successo del 79%.
- "polinomial" : "*-t 1 -d 1*" con successo del 75%

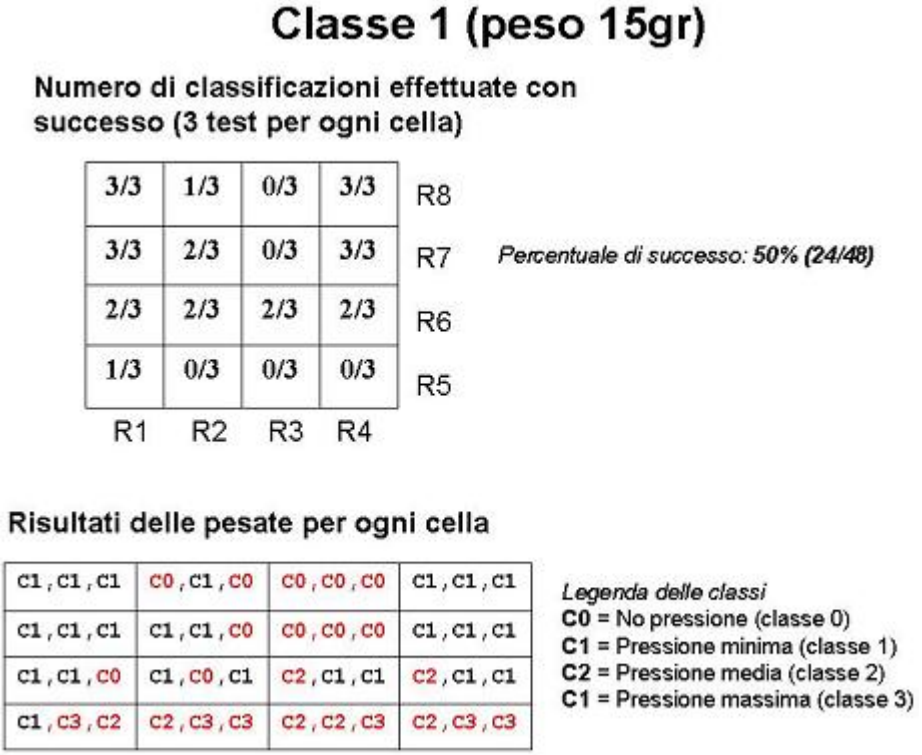
Di seguito sono riportati graficamente, come le seguenti configurazioni classificano i dati:




(Figura 6.10: Nella figura sono riportati i risultati migliori ognuno dei kernel utilizzati, dopo l'addestramento )

Esaminando le configurazioni così ottenute, è evidente che il kernel *Radialbasis* approssima in modo migliore il comportamento delle tre diverse classi. In particolare, si adatta alle sensibilità delle resistenze che compongono il sensore. Va fatta una considerazione sulla configurazione con migliore successo, dall'immagine è evidente che tale configurazione si è adattata all'esempio di training, infatti è presente una zona di overfitting. Mentre, sia nella configurazione con kernel Sigmoid che con kernel Polinomial, la classe 0 (cioè nessuna pressione sul sensore) ha una zona molto ampia, riducendo ls percentuale di successo per la classe 1.
Per cui, la configurazione scelta per i test sul campo è la "*Radialbasis: -c 1*

*-g 50*", che risulta essere la migliore per i nostri scopi. In generale da tale addestramento, si vede che la classe dominante è la classe 3 (pressione massima), questo può essere dovuto ad una non precisa distribuzione dei dati tra le classi.
Il numero di prove per ogni test è di 48 pressioni, distribuite equamente su tutta la superficie del sensore. Il sensore viene visto come una matrice 4x4 dove per ogni cella, vengono effettuato tre pressioni. Di seguito sono riportati i risultati ottenuti:




(Figura 6.11)

Dai risultati ottenuti, si nota che l'errore commesso è del 50%. Occorre tener presente che gli errori sono concentrati su una zona in particolare, ovvero lungo la resistenza R5. L'errore con maggiore frequenza è quello di confondere una "pressione minima" (classe 1), con un di maggiore intensità (classe 2 e 3). Mentre ci sono delle zone in cui la pressione non è stata rilevata (celle R3-R7, R3-R8), per cui hanno contribuito ad aumentare l'errore nei test.

## Classe 2 (peso 25gr)

**Numero di classificazioni effettuate con successo (3 test per ogni cella)**

| | | | | |
|---|---|---|---|---|
| 2/3 | 1/3 | 0/3 | 2/3 | R8 |
| 2/3 | 2/3 | 0/3 | 3/3 | R7 |
| 2/3 | 2/3 | 2/3 | 2/3 | R6 |
| 1/3 | 1/3 | 1/3 | 0/3 | R5 |
| R1 | R2 | R3 | R4 | |

*Percentuale di successo:* ***48% (23/48)***

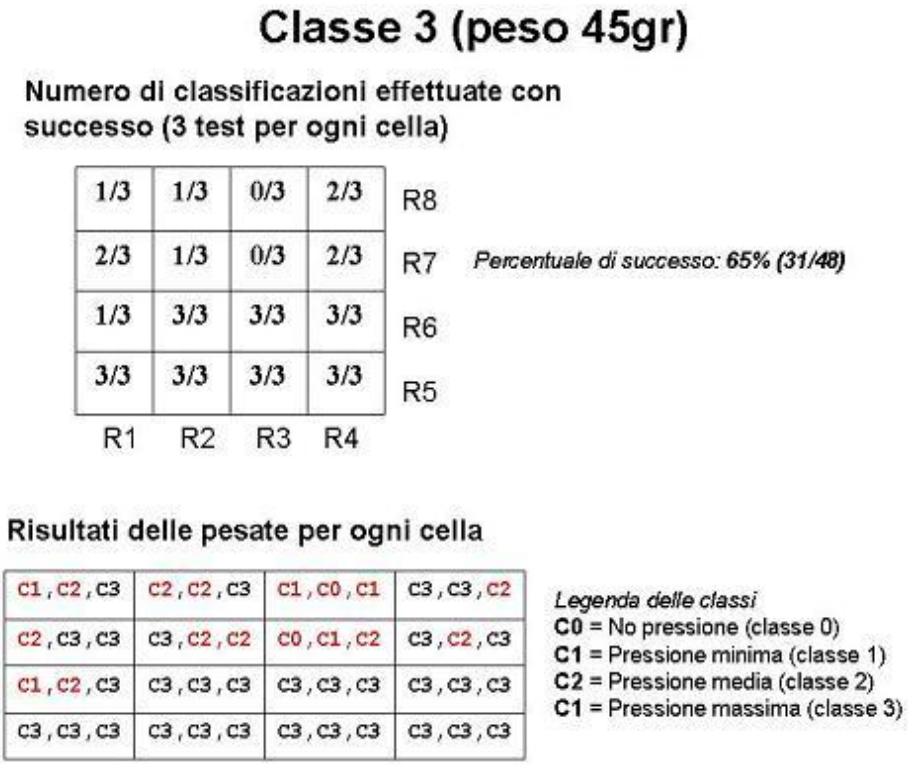
**Risultati delle pesate per ogni cella**

| | | | |
|---|---|---|---|
| C0,C2,C2 | C1,C2,C0 | C0,C0,C0 | C1,C2,C2 |
| C1,C2,C2 | C2,C2,C1 | C0,C0,C1 | C2,C2,C2 |
| C1,C2,C2 | C2,C0,C2 | C2,C2,C3 | C1,C2,C2 |
| C2,C3,C3 | C2,C3,C3 | C3,C3,C3 | C3,C3,C3 |

*Legenda delle classi*
**C0** = No pressione (classe 0)
**C1** = Pressione minima (classe 1)
**C2** = Pressione media (classe 2)
**C1** = Pressione massima (classe 3)

(Figura 6.12)

Anche in questo caso, l'errore commesso è alto (il 52% dei test), comunque gli errori riportati sia per quanto riguarda la zona sul sensore, che la loro tipologia, sono molto simili a quelli riscontrati nel test precedente.




(Figura 6.13)

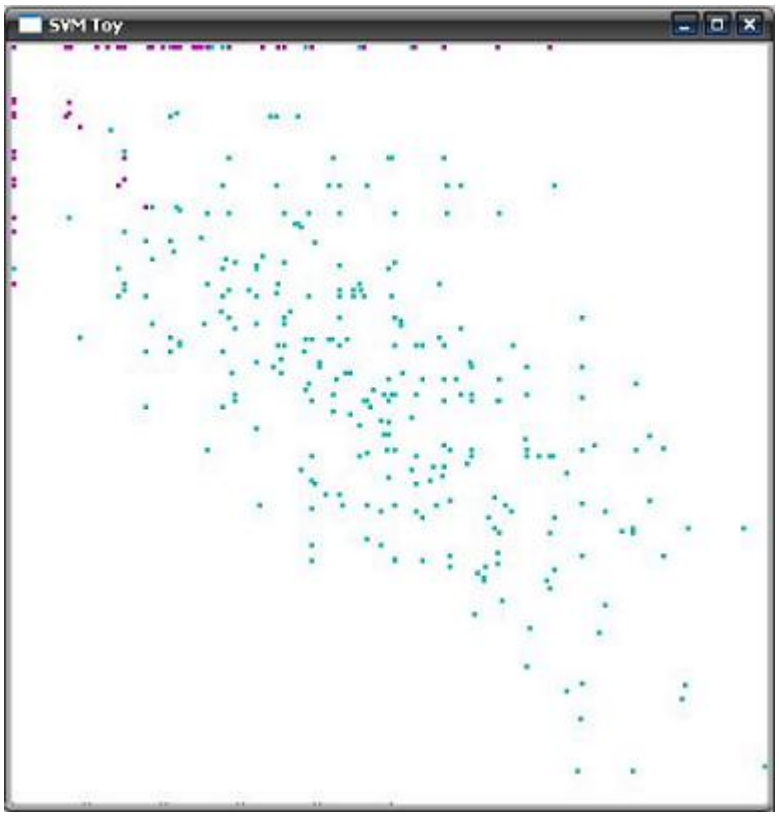
In questo caso, l'errore commesso è inferiore rispetto ai test precedenti, questo perché gli errori precedenti lungo la resistenza R5 (orizzontale in basso), appartengono ad una classificazione di tipo 3 (cioè peso di 45g), come in questo caso.

## 6.4 Test : sfioramento

Per quanto riguarda l'iterazione di tipo "sfioramento", non è stato utilizzato nessun peso, questo perché la superficie del silicone ha una consistenza "quasi ruvida", che si avverte molto con, appunto sfiorando con le dita la sua superficie. Quando viene utilizzato un oggetto metallico per sfiorare la superficie, la variazione del sensore è molto inferiore. Quindi è stato scelto di eseguire lo sfioramento "con le dita", cercando di coprire tutta la superficie del sensore, utilizzando una pressione variabile, come se si stesse "accarezzando" una superficie.

### 6.4.1 Addestramento e test

Di seguito sono riportati i dati ottenuti per il training set:

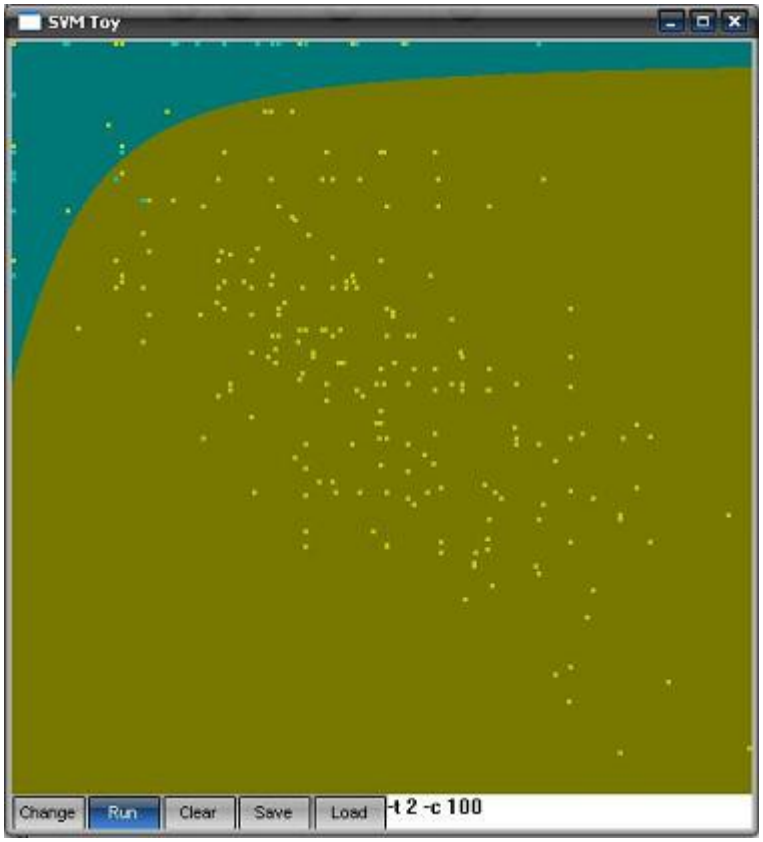



(Figura 6.14: Training set per la classificazione dell'iterazione col sensore di tipo "sfioramento" (classe 4) )

Dalla figura risulta visibile la divisione tra le due classi, inoltre i dati del training set occupano principalmente la zona diagonale del piano, questo dovrebbe facilitare la loro classificazione.
Di seguito viene riportato il risultato del training:

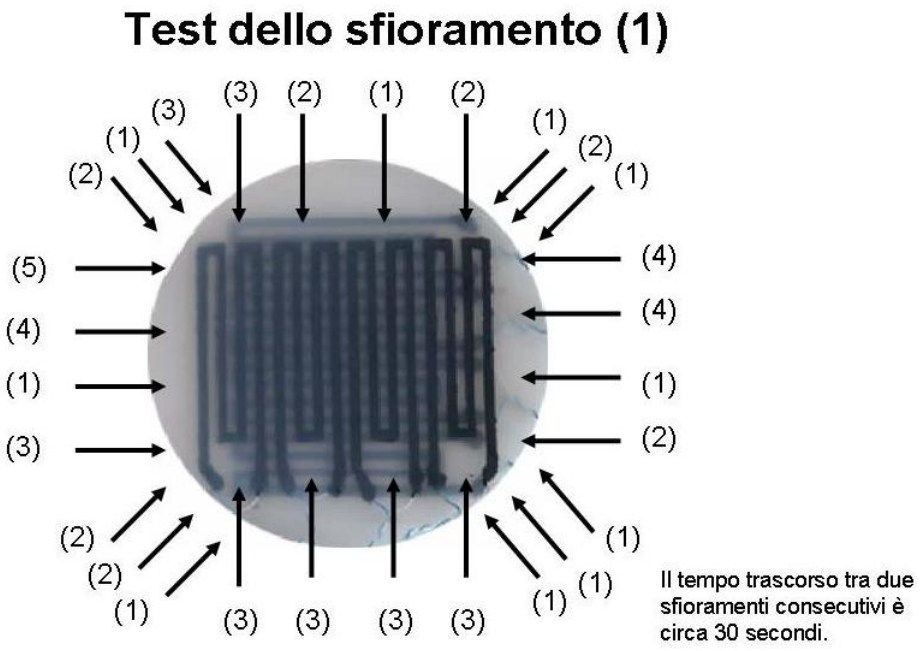
•

In generale tutti i kernel riescono ad effettuare una buona classificazione, dopo un'analisi dei kernel con percentuale più alta di successo è stato deciso di scegliere la configurazione: "Radialbases -c 100", che presenta un successo del 99%, sui dati del predict. Di seguito viene riportato il modo in cui i kernel classifica i dati:



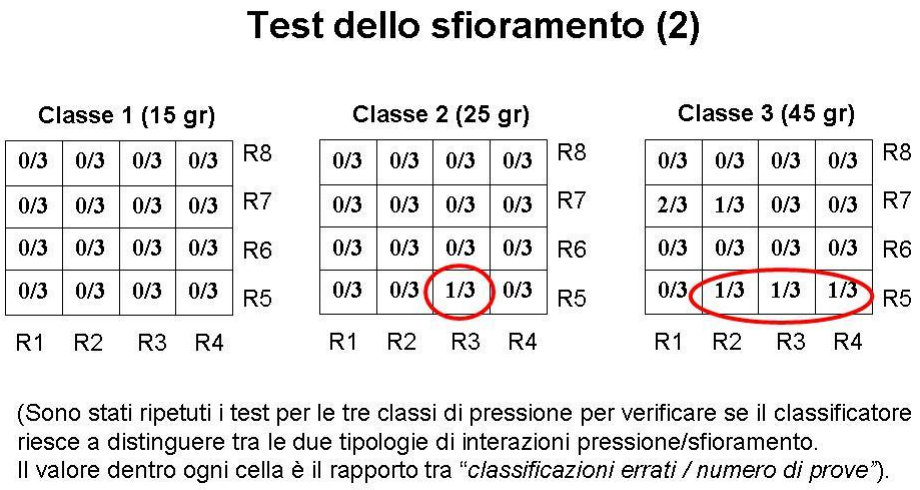

(Figura 6.15: Classificazione fatta dal kernel Radialbases sui dati del file train.txt )

Ad una prima analisi, il risultato atteso dal test dev'essere molto buono, perché il margine tra le due classi è molto ampio.
Il test condotto è diviso in due parti, la prima dove si verifica la percentuale di successo nel classificare gli "sfioramenti", ed una seconda dove vengono effettuale le pressioni con i tre valori di peso (come per i test precedenti) e, verificato l'errore nel confondere un'interazione di tipo pressione con lo sfioramento. Di seguito sono riportati i risultati dei test:




(Figura 6.16: Risultati dei test sulla classificazione dello "sfioramento" )

Il numero medio di sfioramenti necessari per essere riconosciuti dal classificatore è di 2.21. Tale valore, sarà soggetto a variazione è solo un indicazione su quanto è necessario sollecitare il sensore prima di avere una risposta. *É* anche un valore atteso da questo test, perché durante la fase di processing dei dati (nella board utilizzata), è presente il modulo che effettua la media esponenziale con un parametro temporale di 20 instanti di tempo. Infatti, come vedremo nel prossimo risultato, la variazione prodotta da una pressione, non viene avvertita dal classificatore, questo perché la durata della variazione della pressione è molto inferiore a quella prodotta da uno sfioramento.



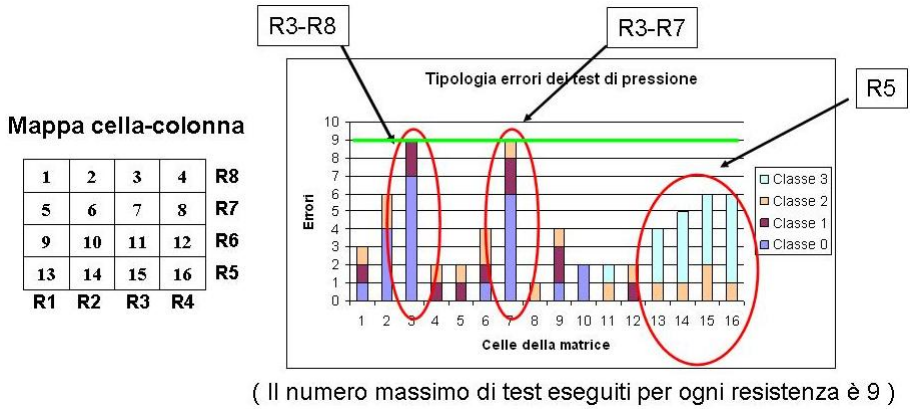

(Figura 6.17: Risultati dei test per quanto riguarda l'errore di classificazione, nel distinguere tra le due tipologie d'interazione con il sensore)

I risultati ottenuti solo molto buoni, su infatti su un totale di 144 test, solo 4 pressioni sono state confuse per "sfioramento", da parte del classificatore.

## 6.5 Analisi risultati dei test

I risultati dei test sono per un certo verso sia positivi che negativi, infatti per quanto riguarda la classificazione dello "sfioramento", la percentuale di successo è molto elevata. Invece per quanto riguarda i test sulla pressione, i risultati non sono buoni. Occorre fare una riflessione proprio sui dati relativi ai test sulla pressione: gli errori di classificazione sono concentrati in zone ben precise del sensore, questo sta ad indicare due motivazioni a tale comportamento:
1) I dati del training set contenevano poche informazioni relative a queste zone, quindi la SVM ha dato poco peso a questi dati. In questo modo la classificazione risulta buona per le zone con più punti nel training set.
2) Le resistenze interne al prototipo di sensore, non hanno tutte la stessa sensibilità, infatti i risultati in corrispondenza delle celle R3-R7 ed R3-R8 sono sempre errati, e la risposta del classificatore è spesso la "classe 0" (ovvero assenza di pressione). Inoltre i test sono stati eseguiti dopo circa un giorno la raccolta dei dati per il training set. Questo è perché sono state provate diverse configurazioni di SVM, ed in fine scelta la migliore. Questa ricerca

del “miglior test”, è durata un paio di giorni, durante i quali sono cambiate le condizioni ambientali di utilizzo, mentre le impostazioni del software (board non sono state modificate).
Ripetendo alcuni test sulle resistenze, adattando la il processing dei dati alla nuova sensibilità della pelle si sono riusciti ad avere risultati migliori dei precedenti. Di seguito riporto una tabella contenete gli errori di classificazione per ogni cella:

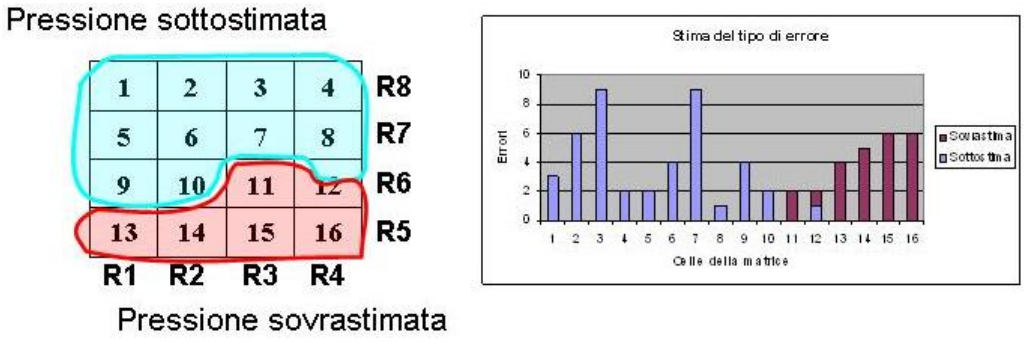



(Tabella 6.18: Tabella errori dei test di pressione, per ogni cella della matrice resistiva del sensore)

Dalla tabella soprastante è evidente la presenza di errori ben precisi per alcune zone del sensore, quali quelle citate in precedenza R3-R7 e R3-R8. Dove la percentuale di errore è del 100%, metre lungo la resistenza R5, è visibile una sensibilità molto elevata alla pressione, perchè gli errori commessi appartengono in maggioranza alla classe 3 (pressione massima di 45 gr). Infatti l’errore lungo R5 è elevato nei primi due test, mentre nullo per quanto riguarda l’ultimo test (classe 3, pressione massima di 45 gr). Se provassimo ad non considerare gli errori nelle zone R3-R7,R3-R8, lungo R5, otterremo i seguenti risultati:

- test classe 1: numero errori: 6/30, successo: 80%.
- test classe 2: numero errori: 9/30, successo: 70%.
- test classe 3: numero errori: 11/30, successo: 63%.
- totale: numero errori: 26/90, successo: 71%.

I risultati dei test migliorano sensibilmente, l'unico a peggiorare di poco è quello relativo alla classe 3.
Un altro aspetto importante, è capire se l'errore commesso dal calssificatore, riguarda una sovrastima della pressione, oppure una sottostima della pressione applicata nel test.




(Tabella 6.19: Tabella errori dei test di pressione, per ogni cella della matrice resistiva del sensore)

L'andamento generale del classificatore, è quello di sottostimare la pressione applicata nella parte superiore, e sovrastimare la pressione applicata nella parte inferiore. La sottostima può essere migliorata, eseguendo di nuovo il training aumentando il numero dei punti relativi alla classe 1 e 2 (15 gr e 25 gr), rispetto a quelli della classe 3 (45 gr). In questo modo, si darà più peso alle prime due classi rispetto alla terza, aumentando la sensibilità per tali classi. Un altro modo per risolvere il problema è quello di modificare il processing dei dati, cercando di distribuire meglio i valori su tutta la scala da 0 a 5000 (nel file di training per la SVM da 0 a 1).

# Capitolo 7

# Conclusioni

Argomenti da discutere:

- problematiche HW incontrare
- sviluppi futuri sul sensore (aumento numero resistenze e dimensione)
- visto che il sensore modifica la sua sensibilità in base alla temperature, sviluppare un modulo che gestisca questa situazione adattando, ad esempio un modulo filtro, in base alla temperatura del sensore.
- possibilità di mettere il classificatore sviluppato direttamente sulla scheda wildfire e di far giare il classificatore li sopra.
- possibili miglioramenti da apportare al software.

# Bibliografia

[Ber82] D.P. Bertsekas. *Constrained Optimization and Lagrange Multiplier Methods*, 1982. Academic Press, New York.

[CL01] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. Software presente all'indirizzo `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[Eri06] Erica Paternostro. Ottimizzazione e caratterizzazione di un ensore per la valutazione del flusso tattile. Master's thesis, Facoltà di Ingegneria, Corso di Laurea in Ingegneria Biomedica, 2006.

[Inc04] Intec Automation Inc. User Manual: WildFire 32−bit Microcontroller Version 1.1 , 2004. Ulteriori informazioni all'indirizzo `http://www.steroidmicros.com/micros/`.

[Joa98] T. Joachims. *Making large-scale support vector machine learning practical*, 1998. Academic Press, New York.

[Vap95] V. Vapnik. *The Nature of Statistical Learning Theory*, 1995. Springer-Verlag New York Inc.

[Vap98] V. Vapnik. *Statistical Learning Theory*, 1998. John Wiley, New York.

...