

ALMA MATER STUDIORUM
UNIVERSITÀ DEGLI STUDI DI BOLOGNA
SEDE DI CESENA

Seconda Facoltà di Ingegneria con sede a Cesena
Corso di Laurea Specialistica in
Ingegneria dei Sistemi e delle Tecnologie dell'Informazione

Using an Evolving Neural Network for the WLAN Location Determination (Localizzazione Indoor in WLAN attraverso Rete Neuronale Evolutiva)

Tesi in
LABORATORIO DI APPLICAZIONI E SERVIZI ICT

Relatore

Ing. Pedro Pablo González Pérez

Presentata da

Claudio Buda

Sessione II
Anno Accademico 2004/2005

Indice

Introduzione	7
1 Localizzazione in WLAN	9
1.1 Wireless Local Area Network	10
1.2 Tecniche	12
1.2.1 Time Of Arrival	12
1.2.2 Time Difference Of Arrival	13
1.2.3 Angle Of Arrival	13
1.2.4 Received Signal Strenght Indication	13
1.2.5 Proxymity	14
1.2.6 Scene Analisys	14
1.3 Perchè e come effettuare tecnica di SceneAnalisys	15
1.4 Tecnica deterministica VS Tecnica probabilistica	16
2 I lavori precedenti	19
2.1 Da cosa si parte?	19
2.2 Tesi Bazzoli	19
2.3 Tesi Gramellini	20
2.4 Tesi Zamagni	20
2.5 Quale strada?	21
3 La tecnica delle reti SOM e delle reti con paradigma di apprendimento competitivo	23
3.1 Le reti neuronali	23
3.2 I modelli di reti neuronali competitive	24
3.3 Le reti Winner-take-all	25
3.4 La rete SOM	28

3.5	Le reti che evolvono dinamicamente	30
3.5.1	Il modello di Sanchez et al. allievi	31
3.6	Come utilizzare l'approccio delle reti competitive in un problema di localizzazione	36
4	Rete Neuronale: Costruzione mappa radio e localizzazione	37
4.1	Analisi del problema	38
4.1.1	Struttura	40
4.1.2	Comportamento	45
4.1.3	Interazioni	48
4.2	Progetto	49
4.2.1	Rete	50
4.2.2	Neurone	63
4.2.3	Pattern	66
4.2.4	Componenti del sistema	67
4.3	Architettura finale	76
4.4	How to use	77
5	Prelevamento di pattern di potenza	81
5.1	Caratteristiche dell'ambiente	81
5.2	La tecnica di prelevamento	81
5.3	L'applicazione	82
5.3.1	Analisi	82
5.3.2	Progetto	83
5.3.3	Prodotto - How to use	91
6	Testing del sistema di localizzazione	93
6.1	Use case: Edificio universitario	94
6.2	Analisi del funzionamento della rete	95
6.2.1	Primo Test: Piano terra dell'edificio	95
6.2.2	Secondo Test: Primo piano dell'edificio	102
6.2.3	Soluzioni proposte	115
6.2.4	Terzo Test: Secondo piano dell'edificio	121
6.3	Paragoni con altri modelli	123

7	Il servizio di localizzazione	127
7.1	Use cases	127
7.1.1	Localizzazione personale	127
7.1.2	Localizzazione di altri utenti	128
7.2	Soluzione 1: Servizi costruiti su chiamate da remoto	129
7.2.1	Localizzazione personale	129
7.2.2	Localizzazione di altri utenti	130
7.3	Soluzione 2: Servizi costruiti su TuCSoN	132
7.3.1	L'infrastruttura TuCSoN	133
7.3.2	Agenti del servizio	136
7.3.3	Centri di tuple identificati	142
8	Conclusioni	145
8.1	Considerazioni finali	145
8.2	Privacy	146
8.3	Ringraziamenti	149
	Bibliografia	150

Introduzione

Tanto si è già parlato di localizzazione, sia essa outdoor, sia essa indoor. Per la prima, il sistema GPS è già una collaudata soluzione al problema, mentre per gli ambienti chiusi le tecniche sono variegate, e nessuna ancora ha preso il sopravvento a causa dei problemi che sorgono a lato pratico. In questa tesi ci si muoverà nella direzione di una di queste tecniche, quella dello Scene Analysis cercando di apportare vantaggi attraverso l'uso di reti neurali che stanno pervadendo gran parte dei settori legati all'informatica e all'intelligenza artificiale grazie al loro dinamismo e alla loro capacità di adattarsi ai rami scientifici più svariati.

Partendo da un progetto di sviluppo orizzontale, che fornisce un'applicazione di reti neurali con linguaggio ad alto livello, questa trattazione attraverserà verticalmente tutti gli strati del modello ISO/OSI per permettere di fornire un sistema funzionante e testabile.

Seguendo la linea temporale di sviluppo attraverso una metodologia a spirale, perciò tornando sulle fasi precedenti in seguito ad eventuali difficoltà riscontrate in quelle successive, verranno esposti ragionamenti e dettagliati resoconti del progetto.

Il primo step sarà quello di analizzare ciò che la comunità scientifica ha già sviluppato e i contributi che già altri elaborati inerenti all'argomento hanno consegnato nel tempo. In seguito si deciderà di costruire un sistema applicativo di rete neurale, modificato per localizzare personale universitario attraverso l'ausilio di una rete wireless. L'ambizione maggiore è infatti quella di riutilizzare la rete wireless 802.11 già presente negli ambienti per fare localizzazione, attraverso reti neurali specifiche a risolvere problemi di SceneAnalysis.

A questa fase si sostituirà un'altra a più basso livello, praticamente fisico,

per prelevare le potenze ricevute dai dispositivi delle rete senza fili che serviranno come input alla rete neuronale. La prelevazione di potenze incontrerà diversi ostacoli legati alla specificità dei dispositivi, vincolando il sistema alle caratteristiche tecniche dell'attrezzatura hardware e dell'ambiente stesso.

Terminata la costruzione del sistema sino qui descritto, una trattazione analizzerà i problemi sorti (causa di una difficile imposizione di uno standard per localizzazione indoor) grazie ad esperimenti effettuati direttamente in ambienti predisposti. Il tempo e le capacità determineranno le soluzioni applicabili nel progetto lasciando una porta aperta ad eventuali sviluppi futuri.

L'ultima fase, quella più ad alto livello, consisterà nel costruire un sistema a servizi, utilizzato solo parzialmente, al fine di localizzare personale, ma espandibile per fornire altri strumenti fruibili da parte degli utenti. Il servizio dovrà perciò affrontare il mondo del "distribuito" nel quale dovrà essere immerso per fornire la capacità di localizzare dispositivi spesso accompagnati da utenti.

Attraverso questo cammino si vuole mostrare come nasce ed evolve un progetto ingegneristico capace di utilizzare idee innovative sino ad avere uno strumento performante e funzionante con numerose prospettive di evoluzione.

Capitolo 1

Localizzazione in WLAN

Tanti sono gli strumenti a disposizione per effettuare localizzazione di utenti, ovviamente uno strumento che facilita la cosa risulta un apparato tecnologico che li rappresenti. Infatti evitando lo scanning visuale o di altri parametri dell'ambiente per carpire la presenza di una persona, il fatto che ognuno sia identificato da un apparecchio semplifica la sua localizzazione mediante la possibilità di comunicare, attraverso l'etere, con l'apparecchio stesso. Infatti se il dispositivo fosse collegato tramite fili, la localizzazione sarebbe alquanto limitata. L'utilizzo di trasmissioni via etere permettono invece di lasciare completa libertà all'utente e di determinare la sua posizione senza invasione (necessita però che l'apparecchio sia costantemente vicino all'utente). Niente di meglio se non un assistente personale o un pc utilizzato per altri scopi (lavoro, svago..) che in sordina permettono la propria localizzazione. Questa elaborazione ovviamente deve essere voluta dall'utente il quale attraverso gli stessi strumenti può facilmente visualizzare informazioni riguardanti altri utenti nella stessa situazione.

Questo capitolo mira a chiarire come può avvenire la localizzazione attraverso il riutilizzo di strutture preesistenti come una rete wireless utilizzata per collegamento ad internet senza fili. Inoltre verranno identificate le tecniche utilizzabili e le specifiche di quella scelta da utilizzare.

1.1 Wireless Local Area Network

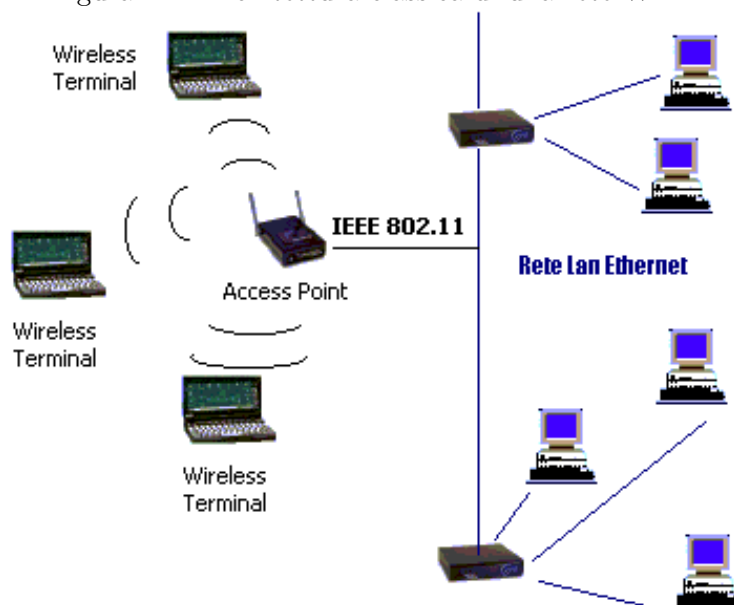
Prima di definire come riutilizzare una Wireless Local Area Network (WLAN) all'interno di un edificio per fare della localizzazione degli utenti, o meglio dei dispositivi che gli utenti utilizzano per formare la rete, si cercherà di descrivere meglio questo tipo di rete citando una fonte.

"Con il termine WLAN (Wireless Local Area Network) si indica un sistema flessibile di comunicazione tra apparecchiature elettroniche che utilizza onde radio al posto delle tradizionali infrastrutture di cavi cablati: una WLAN può essere inquadrata come un'estensione, o meglio un'alternativa, di una LAN cablata. Le WLAN usano onde elettromagnetiche (radio o infrarosso) per comunicare informazioni da un punto ad un altro senza sfruttare nessun collegamento su portante fisico: le WLAN trasmettono e ricevono dati via etere, minimizzando il bisogno di collegamenti cablati, combinando cos'è la connettività con la mobilità dell'utente. Oggi le WLAN sono riconosciute più estesamente come un'alternativa di connettività per un gran numero di utenti.

Solitamente, dove questo sistema è utilizzato, rappresenta l'ultimo anello di congiunzione fra la rete e l'utente finale. Di conseguenza possiamo vedere come un'infrastruttura di tipo Wireless ed una di tipo Wired (cablata) possano coesistere nella stessa rete senza problemi; anzi, questo è addirittura consigliato, in quanto si può in tal modo beneficiare dei vantaggi che i due sistemi possono portare.

Inoltre, la possibilità di utilizzare una WLAN diventa essenziale in quegli ambienti dove è impraticabile, per diversi motivi, l'implementazione di una tradizionale rete cablata. Ad esempio per questioni di costi, o per motivi di non-intrusione in un ambiente preesistente. Il vasto sviluppo del networking nel mondo degli affari e la crescita enorme dei servizi Internet e online sono testimonianze forti dei benefici portati dalla condivisione dei dati e delle risorse. Con le WLAN gli utenti possono accedere ad informazioni condivise senza cercare un luogo per collegarsi in rete

Figura 1.1: Architettura classica di una rete WLAN



e gli amministratori di rete possono implementare o ingrandire una rete senza installare o trasportare fili.” [1]

Questa breve trattazione cita le caratteristiche del protocollo 802.11 standardizzato a livello internazionale dall’Institute of Electrical and Electronic Engineers (IEEE) per l’implementazione di reti WLAN. Di seguito vengono citati gli elementi principali dell’architettura.

“Una 802.11 WLAN è basata su una architettura cellulare in cui l’area dove deve essere distribuito il servizio viene suddivisa in celle proprio come accade nei sistemi di distribuzione per servizi di telefonia GSM. Ciascuna cella (chiamata Basic Service Set o BSS) è controllata da una stazione base denominata Access Point, o più semplicemente AP. Gli AP sono bridge che collegano la sottorete wireless con quella cablata, mentre i Wireless Terminal (WT) sono i dispositivi che usufruiscono dei servizi di rete.

Gli AP possono essere implementati in hardware (dispositivi dedicati), ma anche in software, appoggiandosi, ad esempio, ad

un PC o notebook, dotato sia dell'interfaccia wireless, sia di una scheda Ethernet. Un singolo AP può servire un numero limitato di utenti (tra i 15 e i 50 a seconda del servizio richiesto) e può coprire, a seconda della sua posizione, della topologia della struttura e delle caratteristiche dell'ambiente, un'area che va dai 30 ai 300 metri. L'AP (o l'antenna connessa ad esso) è montato solitamente in alto, ma può essere montato in qualsiasi posizione, fino al raggiungimento della copertura radio desiderata.

I WT possono essere di qualunque tipo: notebook, palmari, PDA, cellulari, apparecchi che interfacciano standard IEEE 802.11. In genere i WT a cui si farà riferimento saranno PC portatili, che per funzionare in sistema WLAN necessitano di una PC Card PCMCIA (o Wireless Card). Essa è una scheda che permette all'utente di accedere alla rete wireless con un sistema del tutto simile a quello di una rete cablata; è composta da un apparato ricetrasmittente, da un'antenna e dall'hardware necessario a rendere disponibile l'interfaccia tra la parte di sistema operativo dedicato alla rete (Network Operative System o N.O.S.) e l'etere, il tutto in modo completamente trasparente all'utente. Queste schede sono inoltre le principali componenti degli AP, che hanno quindi un funzionamento molto simile a quello di un normale utente.” [1]

1.2 Tecniche

Le seguenti tecniche vengono utilizzate sia in ambiente outdoor che in ambienti indoor, attraverso misure di parametri della rete. Queste saranno descritte in modo discorsivo non essendo, questa parte tecnica, l'obiettivo della tesi.

1.2.1 Time Of Arrival

Questa tecnica si basa sulla misura del tempo impiegato al segnale ad arrivare dal trasmettitore al ricevitore. Ovviamente sincronizzando adeguatamente ogni dispositivo grazie al tempo percorso alla velocità della luce del

segnale si giungerà alla determinazione della distanza fra dispositivo utente e le basi (nel nostro caso gli access point) sparse nell'ambiente.

La tecnica è altamente sconsigliata in ambienti outdoor dove i segnali sono affetti da multipath a causa dei rimbalzi provocati dagli ostacoli che si interpongono fra trasmettitore e ricevitore. E' invece preferita come tecnica per dispositivi in linea di vista (LOS), ovvero che non hanno oggetti all'infuori dell'aria lungo la linea ideale più breve che li separa.

1.2.2 Time Difference Of Arrival

Questa tecnica, molto simile alla precedente, misura la differenza fra i tempi misurati dal terminale mobile relativi ai segnali provenienti da tre stazioni fisse. Basterà calcolare due differenze fra i tempi di arrivo dei tre segnali che ecco, una volta conosciuta la posizione delle stazioni basi, sarà facile determinare la posizione del mobile.

Anche se più vantaggiosa rispetto alla precedente non obbligando a sincronizzare il terminale mobile con le stazioni basi, che dovranno invece esserlo, la tecnica soffre del multipath e quindi è sconsigliata per ambienti chiusi.

1.2.3 Angle Of Arrival

Conoscendo l'angolo di arrivo del segnale trasmesso da più basi, due per la localizzazione lungo una retta, tre per una localizzazione puntuale, è possibile utilizzare questa tecnica molto efficiente ma costosa. Infatti per determinare l'angolo di arrivo di un segnale si necessita di un'apparecchiatura piuttosto sofisticata per ricevere il segnale ed elaborare mediante antenne predisposte a tale scopo i dati necessari.

Oltre alla complessità, gioca negativamente il fattore della distanza, che peggiora i risultati per non parlare del multipath che cambia totalmente la correttezza dell'angolo di arrivo se non si è in presenza di segnali in LOS (Line Of Sight).

1.2.4 Received Signal Strenght Indication

La tecnica si basa sulla misura del RSSI (Received Signal Strenght Indication), ovvero la potenza ricevuta dalle stazioni base. Conoscendo la legge

che lega l'indebolimento della potenza nell'ambiente alla distanza che separa trasmettitore e ricevitore è possibile determinare quest'ultima. La legge presa in considerazione è comunque da considerare valutabile facilmente in spazi aperti senza la presenza di ostacoli che degradano il valore di potenza, e cammini multipli del segnale che aumentano la distanza che il segnale percorre.

Queste varianti vanno ad incidere enormemente su questa tecnica, tanto da rendere impossibile non valutarle in un ambiente chiuso; inoltre ogni ambiente chiuso determina una variazione univoca della legge che lega potenza e distanza. Bisognerebbe perciò fare un attento studio del luogo prima di utilizzare questa tecnica per determinare la corretta relazione potenza - distanza (si consideri però che anche solo il passaggio di un utente può avere un effetto su di essa). Risulta troppo complicato e irrealizzabile utilizzare questa in un ambiente chiuso in continuo dinamismo.

1.2.5 Proximity

Si potrebbe anche determinare grossolanamente la posizione di un oggetto attraverso la rilevazione di un segnale che non sarebbe avvenuta se l'oggetto fosse stato più distante. Si pensi infatti al collegamento di un portatile alla rete wireless ad un access point. Questo determina la localizzazione dell'utente nell'area coperta dall'access point. Oppure a sensori di contatto quali touch screen o sensori di vicinanza come RFID o lettori di immagini. Tutti questi apparati consentono una grossolana individuazione dell'oggetto ma effettivamente non permettono di monitorarne gli spostamenti se non attraverso l'uso contemporaneo di più dispositivi che però risulterebbe troppo costoso per avere un dettaglio sufficiente della posizione rilevata.

1.2.6 Scene Analysis

L'ultima tecnica che sarà quella scelta per una serie di motivi economici, strutturali e tecnici, permette la localizzazione dell'utente attraverso il paragone fra potenze misurate in un ambiente che non ha subito notevoli variazioni nel tempo che è trascorso tra le due rilevazioni.

La tecnica infatti prevede la misurazione di tante potenze nei vari punti di un ambiente (fase offline). Maggiori saranno i punti dove effettuare le

rilevazioni, maggiore sarà il dettaglio di localizzazione. In seguito, portandosi su un punto del luogo in esame, sarà possibile, rilevando di nuovo una potenza (fase online), conoscere la posizione all'interno dell'area d'interesse, confrontandola con quelle precedentemente rilevate. Fra queste, una avrà "distanza" minore delle altre, perciò sarà deducibile che l'utente si trova in prossimità dello stesso punto dove essa è stata prelevata durante la fase offline.

1.3 Perchè e come effettuare tecnica di SceneAnalysis

Come è stato già anticipato è stata prediletta questa tecnica per motivi di riutilizzo delle risorse, quali gli stessi access point utilizzati per creare la WLAN (senza doverli aumentare di numero per ottenere una adeguata precisione), e per motivi economici in quanto non si necessita di alcuna hardware aggiuntivo per l'utente, il quale riutilizzerà la stessa scheda con cui si collega alla rete wireless. Un ulteriore motivo è quello della presenza di multipath fading in un ambiente chiuso.

Lo scene analysis si divide perciò in due fasi, la prima, definita anche fase "offline", per registrare le potenze ricevute dai singoli access point, la seconda, la fase "online", dove si propone un vettore di potenze ricevute (pattern) da confrontare con i vettori registrati nella precedente fase.

La fase offline empirica potrebbe essere sostituita dalla stesura di un modello analitico che sostituisce le numerose rilevazioni da effettuare nell'ambiente ad ogni suo cambiamento e soprattutto durante la prima serie di rilevazioni di potenze. Un modello analitico determina la funzione densità di probabilità della potenza che viene rilevata da ogni singolo AP.

Si parte da un modello analitico che tiene conto della presenza di multipath all'interno dell'edificio. Questo modello detto di Fading alla Rayleigh non considera che fra le componenti del segnale possa essercene una più significativa (in LOS). Allora per superare questa divergenza si può far ricorso al modello di Fading alla Rice che invece considera, con un coefficiente proporzionale da determinare, l'esistenza di una componente del segnale più

significativa. Infine occorre citare un ulteriore modello, questa volta di tipo deterministico e non probabilistico, che inserisce un fattore di attenuazione di piano (Wall Attenuation Factor) al calcolo della potenza, cioè un coefficiente che considera la diminuzione della potenza apportata dalla presenza di muri. Ovviamente questo fattore sarebbe da modificare da edificio ad edificio, così come i parametri dei modelli statistici.

Questi modelli sono molto utili per evitare la lunga raccolta di dati empirici dei segnali di potenza ma alla fine sono troppo dipendenti dal luogo in cui si utilizzano. Infatti determinare tutti i parametri risulterebbe un lavoro molto più brigosio che rilevare punto per punto la mappa radio ovvero tutte le potenze della fase offline. La mappa radio sarà così determinata. Le scelte che rimangono da fare sarà se mantenere valori deterministici delle potenze rilevate dagli access point in un punto o mantenere tutta la statistica decidendo di non mediare le rilevazioni effettuate in un punto, ma di costruire una funzione densità di probabilità. Si pensi anche alla soluzione di mantenere solamente media e varianza invece che tutta la distribuzione. Questi due metodi di registrazione della mappa radio determinano se utilizzare una tecnica deterministica o probabilistica per effettuare la localizzazione.

1.4 Tecnica deterministica VS Tecnica probabilistica

Come detto nella precedente sezione decidere la tecnica significa avere dati in formato diverso all'interno della mappa radio. La differenza non sta nel prelievo del pattern di potenze per la localizzazione. Questo infatti potrà essere solamente frutto di una media fra pochissimi rilievi per permettere ad un servizio di localizzazione l'efficienza necessaria.

La tecnica deterministica significa avere semplicemente due valori da confrontare tante volte quanti sono i valori della mappa radio. Il primo valore infatti è pescato dalla mappa costruita in precedenza o empiricamente o con il modello di attenuazione di piano, mentre il secondo risulterà quello rilevato per determinarne la posizione. Già la Microsoft, con l'algoritmo RADAR, ha precorso i tempi gettando le basi per la determinazione del

pattern più vicino. La prima versione dell'algoritmo è stata quella NNSS (Nearest Neighbor in Signal Space). Questa non fa altro che calcolare la distanza euclidea fra due vettori di potenza per ogni pattern della mappa radio. Quello a distanza euclidea minore dal vettore da localizzare vince e lo localizza nelle sue vicinanze. Questo algoritmo nasce dal criterio MAP (Maximum A Posteriori) per la determinazione del vettore di potenza più vicino considerando perciò solo la media delle potenze, che risulta infatti il valore deterministico registrato. Sempre da casa Microsoft sono nate numerose varianti, vale la pena citare quella in cui si considera la posizione precedente dell'utente. Infatti tenendo conto di ciò e che l'utente non può spostarsi a velocità elevata, un suo monitoraggio viene facilitato dall'esclusione di posizioni troppo lontane dalle posizioni precedenti.

Le tecniche probabilistiche invece avranno una mappa radio che conterrà la frequenza relativa delle potenze rilevate per ogni access point per ogni punto di prelievo. Se avremo frequenze relative questa mappa sarà stata costruita empiricamente. Se si vuole tutta la funzione densità allora si potranno utilizzare i modelli analitici già citati. Verranno citate in questa tesi due tecniche proposte da un gruppo di ricercatori dell'Università del Maryland. La prima, ovvero quella del Join Clustering, oltre alle informazioni statistiche del segnale, necessita di una clusterizzazione che viene eseguita raggruppando posizioni che hanno alcune caratteristiche comuni, permettendo di ridurre notevolmente lo spazio di ricerca e, quindi, di ridurre i calcoli computazionali richiesti dall'algoritmo di localizzazione. Per la localizzazione si sceglie un numero di access point minimo da considerare, solitamente quello ha potenza maggiore e in seguito si determina, attraverso il criterio ML (Maximum Likelihood) e alcune semplificazioni, quale pattern della mappa radio ha più probabilità di essere corretto (o quello più vicino a quello misurato). Un'altra tecnica, quella della Triangolazione Incrementale, utilizza gli access point in modo incrementale, uno dopo l'altro, fino a quando non si ottiene una stima corretta della posizione che presenta un'accuratezza che sta entro una soglia predeterminata. [2]

In seguito a risultati sperimentali le tecniche probabilistiche danno maggiore accuratezza ai risultati e inoltre il tempo di calcolo è ridotto grazie

all'utilizzo di cluster per le tecniche probabilistiche. Ovviamente le seconde comportano difficoltà maggiore soprattutto nella fase offline dove ad essere registrato dovrà essere un numero notevolmente superiore di informazioni.

Capitolo 2

I lavori precedenti

Questa tesi è caratterizzata da un lungo cammino attraverso gli strati della architettura ISO-OSI partendo da altre tesi e ricerche intraprese ai vari livelli ma mai assemblate, che non forniscono strumenti applicativi per una realizzazione effettiva di servizio completo. Lo scopo principale è quello di fornire un servizio di localizzazione per utenti mobili attraverso una metodologia già definita da precedenti studi [3].

2.1 Da cosa si parte?

Si parte da tre tesi completamente sviluppate in un ambito prettamente comunicazionista ma senza affrontare a pieno la complessità applicativa. Nelle prossime sezioni verranno esplicitati gli apporti che queste tesi hanno dato per l'inizio del progetto in discussione, delle soluzioni proposte e delle evoluzioni future alle quali si è tentato di avvicinarsi.

2.2 Tesi Bazzoli

La tesi che affronta il problema della localizzazione nel suo aspetto più generale e analitico è quella di Francesco Bazzoli [1]. In questa tesi si analizzano varie tecniche e metodologie di localizzazione indoor, attraverso strumenti diversi fra cui la rete WLAN.

L'obiettivo è stato quello di fornire un'adeguata analisi delle opportunità messe a disposizione dalla comunità scientifica per effettuare localizzazione

indoor. La scelta è ricaduta sulla tecnica di SceneAnalysis con un accuratezza maggiore utilizzando algoritmi probabilistici invece che deterministici comunque validi. Inoltre è stato implementato un software su piattaforma Linux per prelevare in maniera ordinata valori di potenza dai singoli access point e poter costruire facilmente la mappa radio sulla quale poter applicare l'algoritmo più idoneo.

2.3 Tesi Gramellini

Questa tesi di Maria Chiara Gramellini [2] si spinge oltre alla presentazione di algoritmi per la localizzazione indoor, infatti mira ad inserire un algoritmo basato su rete neurale, sia esso di stampo deterministico, sia probabilistico. Vengono forniti perciò due modelli di rete neurali comuni nella fase offline, ovvero di memorizzazione della mappa radio, ma diversi in quella online.

Nella tesi vengono ben descritti gli aspetti di una rete neurale e il percorso che porta alla scelta di una rete di tipo Self Organizing Map per la costruzione della mappa radio e di una rete Probabilistic Neural Network per l'attuazione dell'algoritmo probabilistico. Entrambi le reti vengono messe all'opera attraverso la simulazione con valori casuali, dando risultati soddisfacenti ma senza avere la reale concezione di cosa significhi utilizzare effettivamente una rete neurale su un sistema applicativo con valori reali.

L'inserimento delle reti neurali da un apporto non indifferente all'evoluzione della mappa radio in un edificio che cambia, infatti la rete permette attraverso un addestramento di modificare la mappa senza ricorrere a prelevare di nuovo ogni pattern di potenza per ogni punto all'interno dell'edificio.

2.4 Tesi Zamagni

La tesi di Guido Zamagni [4] fornisce un prodotto finito, capace di dare risultati eccellenti su un vasto numero di prove. Infatti viene realizzata la tecnica RADAR così come è stata concepita dai ricercatori Microsoft sviluppando l'applicazione con linguaggio C e carpando i dati sulle potenze attraverso strumenti forniti dal sistema operativo Linux.

In questa tesi viene avvalorata la regola di escludere dal calcolo gli access point con potenze più basse perchè considerati meno stabili. Attraverso una accurata media dei campioni i risultati raggiungono un errore in media di tre metri circa.

2.5 Quale strada?

Questa trattazione che parte dalla costruzione di una rete neuronale estendibile e il più dinamica possibile, mira a sviluppare un servizio di localizzazione utenti utilizzando la tecnica di scene analysis di tipo deterministico lasciando spazio ad evoluzioni probabilistiche future. Per utilizzare la tecnica si sfrutterà la rete neuronale più adatta e adeguata per ottenere i migliori risultati. In ordine temporale si andrà poi a basso livello per costruire un'applicazione che permetta di registrare i valori di potenza su un sistema operativo di più largo utilizzo come Windows. Queste misurazioni serviranno alla rete per costruire la mappa di potenza. Ultimo step la costruzione di un servizio. Il servizio localizzazione sarà costituito dalla rete stessa che potrà fornire agli utenti informazioni sulle posizioni di altri utenti.

Capitolo 3

La tecnica delle reti SOM e delle reti con paradigma di apprendimento competitivo

3.1 Le reti neurali

”Le reti neurali artificiali (RNA) sono sistemi di processamento dell’informazione distribuiti e paralleli ispirati alla struttura e al funzionamento delle reti neurali biologiche.

Questi sistemi di processamento dell’informazione sono concepiti per emulare certe funzioni eseguite dal cervello umano, quali riconoscimento dei patterns, classificazioni, apprendimento, memoria e l’astrazione della conoscenza fra le altre funzioni.”

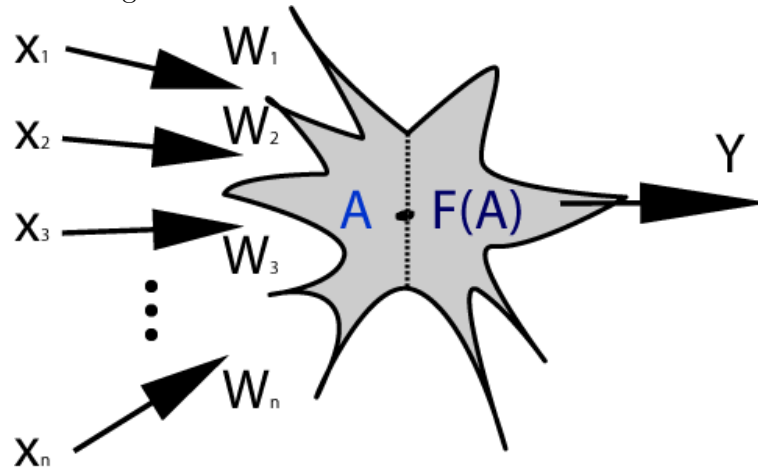
[2]

Queste reti sono formate dall’insieme di neuroni artificiali collegati tra loro in diverse modalità dove ogni collegamento è caratterizzato da un peso per il neurone che sta a valle verso il neurone a monte.

”I neuroni artificiali rappresentano una particolare informazione, cioè, analogamente ai sistemi biologici, lo spazio neurale viene utilizzato per dare una rappresentazione degli eventi esterni.

Vediamo quali sono le grandezze, mostrate in figura, che lo caratterizzano.

Figura 3.1: Architettura del neurone artificiale



Il vettore $X = (X_1, \dots, X_n)$ rappresenta il vettore degli ingressi al neurone che possono essere a loro volta le uscite di altri neuroni quando sono presenti delle sinapsi. Il vettore $W = (W_1, \dots, W_n)$ è il vettore dei pesi, che rappresenta cioè il peso da attribuire alla sinapsi corrispondente. A non è altro che la Funzione di attivazione che è la funzione che stabilisce il livello di attivazione del neurone artificiale come conseguenza degli ingressi ricevuti. In corrispondenza al piano biologico questa funzione identifica gli insiemi dei valori che assume il Potenziale di Membrana a Riposo (PMR) come conseguenza della sommatoria spazio-temporale dei potenziali postsinaptici. Le funzioni più comuni usate sono: funzione identità, funzione rampa, funzione Gaussiana, ecc... $F(A)$ invece è la funzione di propagazione che stabilisce quando il neurone è autorizzato a dare un'uscita, corrisponde quindi alla funzione di uscita.” [2]

3.2 I modelli di reti neurali competitive

Le reti neurali competitive [5] costituiscono una importante variante delle reti neurali artificiali in cui la rete cerca di scoprire caratteristiche e pat-

terns speciali dai dati di entrata, senza che ci sia alcuna supervisione esterna. Questo tipo di rete utilizza algoritmi d'apprendimento senza supervisione per eseguire compiti come:

- *Clustering*. Il clustering (raggruppamento) consiste in raggruppare o dividere un insieme di dati in sottoinsiemi separati (clusters), in modo tale che i patterns appartenenti a uno stesso cluster siano molto simili fra essi, mentre i patterns appartenenti a clusters diversi siano più o meno possibile differenti.
- *Vector quantization*. Questo tipo di compito è molto simile al clustering. Lo spazio di entrata è diviso in diverse regioni connesse. Ciascuna regione viene rappresentata utilizzando un unico vettore. Ciascun pattern nello spazio di entrata appartiene a una regione particolare (quella che possiede il vettore più vicino al pattern).
- *Distribuzione di probabilità*. Il metodo consiste in ottenere una descrizione riassuntiva di grandi volumi di dati estratti da una distribuzione di probabilità. La descrizione potrebbe consistere in un insieme di punti dello spazio di dati, tale che la distribuzione di questi punti costituisca un'approssimazione della distribuzione originale.
- *Estrazione di caratteristiche*. Il metodo consiste in scoprire le caratteristiche più importanti (cioè, quelle che posseggono maggiore variazione) nello spazio di dati. Seguendo questo metodo, patterns appartenenti a clusters differenti possono essere individuati da alcune caratteristiche. L'estrazione di caratteristiche porta a una riduzione della dimensione dei patterns di entrata.

3.3 Le reti Winner-take-all

Le reti winner-take-all costituiscono il principale approccio per eseguire l'apprendimento competitivo. Questo meccanismo presuppone l'utilizzo dello spazio neuronale (strati di neuroni) per la rappresentazione delle categorie, le quali vengono organizzate su questo spazio in una maniera spontanea e dall'interazione con gli eventi esterni. Le principali caratteristiche di questo tipo di rete sono riassunte in seguito:

- La rete neurale è organizzata in strati. Ci sono collegamenti eccitatori fra i neuroni appartenenti ai diversi strati e collegamenti inibitori fra i neuroni appartenenti allo stesso strato.
- I neuroni nello strato di uscita stabiliscono una gara fra loro per rappresentare il pattern di ingresso, ognuno inibendo gli altri con il suo livello di attivazione attuale.
- Dopo la gara, il neurone vincitore è attivato, e il suo vettore di peso sarà aggiustato per avvicinarlo al pattern di ingresso.
- I pesi sinaptici sono aggiornati in accordo ad una regola Hebbiana generalizzata.

In seguito approfondiremo un po' più le idee precedenti.

L'idea base di questo principio di organizzazione è la seguente: i neuroni o insieme di neuroni che rappresentano una porzione dello spazio neurale competono fra essi per rispondere alla presentazione di una particolare categoria di eventi esterni. La gara si basa sui diversi livelli di attività raggiunti dai diversi neuroni, data la presentazione di un evento esterno, nonché in una struttura di inibizione laterale esistente fra i neuroni. Una rete winner-take-all è composta da diversi strati. Le caratteristiche del processamento cambiano a seconda di dove questo si svolge: fra gli strati o all'interno di uno strato particolare.

Quando il processamento prende luogo fra gli strati:

- Ciascun neurone in uno strato inferiore si collega a tutti i neuroni ubicati nello strato immediatamente superiore
- Tutte le connessioni sono eccitatrici
- Il flusso dell'informazione è del tipo feedforward

Il processamento all'interno di un strato tipicamente viene caratterizzato da:

- I neuroni dentro uno strato (tranne nello strato di entrata) si dispongono in clusters, i quali possono o non definire regioni di intersezione fra essi
- Ciascun neurone dentro un cluster si collega con i restanti neuroni del cluster attraverso connessioni inibitrici
- L'attività fluisce in maniera ricorrente fra i neuroni del cluster, finché non sia raggiunto uno stato nel quale solo un neurone è in grado di mantenersi attivo

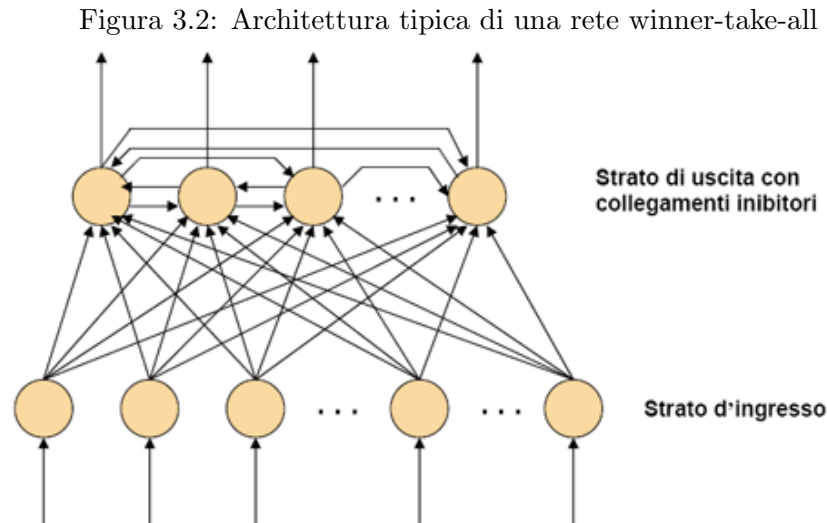
I neuroni vincitori per ciascun cluster saranno quelli che invieranno la loro uscita al prossimo strato. Dunque, il pattern di entrata ad uno strato superiore sarà quello che risulta dai processi inibitori che accadono nello strato immediatamente inferiore.

In una rete winner-take-all l'apprendimento è competitivo giacché il processo di aggiornamento dei pesi si verifica in funzione dei neuroni vincitori. Un neurone "vince" la gara quando risponde con un livello di attività più elevato rispetto agli altri neuroni del cluster alla presentazione di un particolare pattern originato dalle uscite dei neuroni dello strato inferiore. Il vettore di pesi del neurone vincitore è modificato e questa modifica permette che questo neurone risponda ancora con maggiore forza a successive presentazioni dello stesso pattern oppure di patterns simili. Questo fatto porta a una "specializzazione" graduale del neurone nel riconoscimento di patterns particolari.

Il processamento dell'informazione in una rete winner-take-all può essere descritto nei seguenti termini: l'organizzazione raggiunta da ciascuno strato durante il processo d'apprendimento permette che ogni strato neuronale esegua un processo di estrazione di caratteristiche dai patterns che arrivano dallo strato immediatamente inferiore. L'organizzazione gerarchica degli strati garantisce il fatto che le caratteristiche ricavate siano ogni volta più astratte. In altri termini, le caratteristiche comuni scoperte in strati superiori sono ogni volta meno evidenti, quando si considerano i patterns primari presentati nello strato di entrata. Di conseguenza, nell'ultimo strato si raggiunge l'implementazione di un sistema di categorie che segue la struttura

di organizzazione "naturale" degli eventi con i quali interagisce la rete.

La figura 3.2 ne mostra l'architettura tipica.



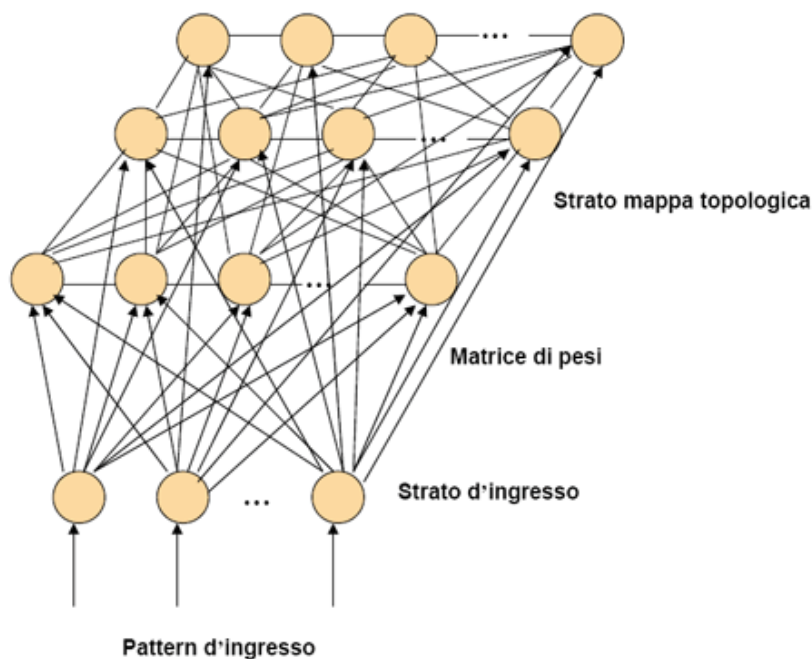
3.4 La rete SOM

Agli inizi degli anni ottanta T. Kohonen [6] [7] propose la più utilizzata delle reti organizzate in funzione della topologia, conosciuta come mappa auto-organizzativa (dall'inglese self-organizing map (SOM)), mappa che preservava la topologia o mappa di caratteristiche auto-organizzative (SOFM). La SOM di Kohonen combina un principio di apprendimento competitivo con una strutturazione topologica dei neuroni, in modo che i neuroni adiacenti tendano a possedere vettori di pesi simili. Nella rete SOM i neuroni tendono a possedere vettori di pesi che prendono le caratteristiche dello spazio vettoriale di entrata. La relazione di vicinanza si traduce nella prossimità fra i neuroni nello spazio Euclideo, pur essendo i vettori di pesi iniziali arbitrari.

La rete SOM di Kohonen è composta da due strati (vedere figura 3.3), un primo strato vettoriale che costituisce lo strato di entrata con dimensione pari a quella dei patterns di ingresso, e un secondo strato a forma di matrice o esagono il quale è lo strato di uscita dove evolve la strutturazione topolo-

gica della rete. Ciascun neurone dello strato di entrata si collega con ciascun neurone dello strato di uscita e la forza di tali connessioni viene data dai vettori n -dimensionali $w_j = (w_{j1}, \dots, w_{jn})$ che caratterizzano ogni neurone dello strato di uscita. A questi pesi vengono assegnati inizialmente valori aleatori.

Figura 3.3: Architettura tipica di una rete winner-take-all



Nella rete SOM i valori dei pesi vengono modificati attraverso il processo di apprendimento. In questo processo ogni vettore di pesi si muove nella direzione del prototipo (o centro) di qualche sottoinsieme di patterns di entrata. L'uscita di ciascun neurone dello strato di uscita rappresenta la vicinanza del neurone al pattern di entrata.

Quando un pattern di ingresso viene presentato nello strato di entrata, l'algoritmo SOM aggiorna il vettore di peso del neurone vincitore ed anche i vettori di pesi dei neuroni che appartengono alla regione di vicinanza topologica definita per il neurone vincitore. L'unico criterio da seguire è la distanza topologica fra i neuroni nello spazio Euclideo. Un'area $N_j(t)$ contiene i neuroni che si trovano ad una distanza topologica $D(t)$ dal neurone j

nel tempo t , dove $D(t)$ diminuisce col tempo. $D(t)$ non si riferisce a distanza Euclidea fra i patterns di ingresso e i vettori di pesi, bensì alla lunghezza del percorso che collega due neuroni dello strato di uscita a seconda della topologia scelta per la rete.

Quando una rete SOM è usata per classificazione di pattern, i pattern d'ingresso sono assegnati a un gruppo di clusters sulla base della similitudine fra loro e i vettori di riferimento dei clusters. Prima di iniziare il compito di classificazione o riconoscimento di pattern, l'utente deve predefinire una topologia di nodi (tipicamente un array rettangolare o esagonale bidimensionale, dove ogni nodo nell'array rappresenta un cluster). All'inizio, vettori di riferimento aleatori sono generati e assegnati a ciascun nodo. Dopodiché, un processo iterativo esegue i seguenti passi:

1. Un pattern d'ingresso è scelto in modo aleatorio e il nodo più vicino a questo, in termini di una distanza, è scelto.
2. Il vettore di riferimento del nodo scelto (nello spazio del pattern d'ingresso) è allora aggiustato, in modo tale che questo diventi più simile al pattern d'ingresso assegnato.
3. I vettori di riferimento degli altri nodi vicini al nodo scelto nell'array bidimensionale (a seconda un criterio di vicinanza) vengono aggiustati.

La tabella riassume le principali caratteristiche delle reti SOM.

3.5 Le reti che evolvono dinamicamente

Il modello di rete proposto da Sanchez et al. nel 1995 [8] permette di descrivere la dinamica di una rete che evolve dinamicamente. Su questo modello è stato ispirato il modello di rete neuronale evolutiva come approccio al problema di localizzazione in ambiente WLAN, descritto nel prossimo capitolo.

In questo tipo di reti non solo i vettori di peso vengono modificati nel processo d'apprendimento della rete. Tipicamente la rete parte da una strut-

Tabella 3.1: Caratteristiche rete SOM

Primo strato di neuroni	Strato d'ingresso
Secondo strato di neuroni	Strato mappa topologica
Pattern di collegamento	Feedforward Connessioni inibitrici fra tutti i nodi della mappa topologica
Tipi di valori d'ingresso	Reali, binari
Funzione NET	Distanza Euclidea
Funzione di attivazione	Sigmoide
Metodo d'apprendimento	Competitivo
Algoritmo d'apprendimento	Algoritmo self-organizing

tura o configurazione minima di neuroni di peso, e man mano che arrivano i pattern di ingresso alla rete, i seguenti processi prendono luogo:

- Creazione di neuroni nascosti
- Creazioni di neuroni di uscita
- Creazione di collegamenti fra i neuroni d'ingresso e i neuroni nascosti
- Creazione di collegamenti fra i neuroni nascosti e i neuroni di uscita
- Eliminazione di neuroni
- Creazione di clusters
- Spostamento di neuroni
- Aggiornamento di altri parametri che caratterizzano i neuroni (ad esempio, il raggio)

3.5.1 Il modello di Sanchez et al. allievi

Il modello proposto da Sanchez et al. è stato sviluppato per affrontare il problema dell'apprendimento e classificazione di pattern appartenenti a classi che non sono linearmente separabili e la cui distribuzione varia nel

tempo, così come per pattern appartenenti ad ambienti stazionari e non-stazionari.

La rete impara a classificare pattern in un modo supervisionato utilizzando per questo compito un insieme di neuroni nascosti che vengono creati e collegati fra loro a seconda il bisogno del problema di classificazione in questione.

L'apprendimento implica la creazione di nuovi neuroni nascosti, l'aggiornamento dei neuroni esistenti nonché dei suoi parametri. Fra i parametri dei neuroni nascosti (chiamati nel modello di Sanchez, centro) si trovano il coefficiente di vitalità e il raggio, tutti e due usati durante la classificazione dei pattern e per decidere sulla permanenza o esclusione del neurone dalla rete.

Il modello neuronale è caratterizzato dalle seguenti proprietà:

- Apprendimento supervisionato
- Apprendimento in tempo reale
- Esistenza di un raggio per definire le regioni di classificazione dei neuroni nascosti
- Adattamento dei neuroni attraverso i suoi parametri
- Incremento controllato del numero dei neuroni nascosti
- Garanzia della convergenza della rete nel processo di classifica
- Apprendimento di patterns appartenenti ad ambienti stazionari e non-stazionari

I parametri della rete neuronale

La rete usa diversi parametri per controllare il suo funzionamento, alcuni di questi sono forniti dall'utente, altri sono derivati dai pattern d'ingresso. Man mano che la rete evolve nel tempo, alcuni di questi parametri verranno modificati.

X: pattern d'ingresso

- R_p : raggio del neurone nascosto
 P : neurone nascosto
 V_p : coefficiente di vitalità del neurone nascosto p
 β_v : costante di decisione di eliminazione di neuroni, usata dal meccanismo di dimenticanza
 g_v : costante di decisione di eliminazione di neuroni, considerando il raggio
 M : numero totale di neuroni nascosti
 PC : classe alla quale appartiene il pattern d'ingresso
 WC : classe che rappresenta il neurone vincitore
 R_o : raggio iniziale
 $d(X,P)$: distanza Euclidea fra il pattern X e il neurone nascosto P
 α : tasso di aggiornamento

Algoritmo della rete neuronale

All'inizio la rete non ha memoria, cioè, non ci sono neuroni nascosti nello strato nascosto. Ci sono soltanto uno strato d'ingresso e uno strato di uscita.

Il numero di neuroni nello strato d'ingresso corrisponde alla dimensione dei pattern d'ingresso, mentre il numero di neuroni nello strato di uscita rappresenta il numero di classi o categorie nelle quali verranno classificati i pattern d'ingresso. I neuroni nascosti e i clusters vengono creati man mano che arrivano i patterns di entrata.

1. Computare la distanza $d(X,P_i)$ e $V_{pi} = V_{pi}/\alpha$, per $1 < i < M$.
Determinare il neurone vincitore P con raggio R_p . Denotare WC la classe del neurone P .
2. Se $PC = \text{"Non classificato"}$, allora il pattern X è assegnato alla classe WC e il processo conclude.
3. Se $WC = \text{"Non classificato"}$, allora un nuovo neurone nascosto è creato e il pattern X verrà associato a questo neurone. Saltare al punto 5.
4. Se $WC \neq \text{"Non classificato"}$, allora:
 - (a) Se $d(X,P) < R_p$ e $WC = PC$, allora aggiornare il neurone P facendo $V_p = V_p * 2$. Saltare al punto 5.

(b) Se $d(X, P) < R_p$ e $WC \neq PC$, allora ridurre R_p . Saltare al punto 5.

(c) Se $d(X, P) \geq R_p$ allora:

- Se è possibile unire P e X, aggiornare $V_p = V_p * 2$. Saltare al punto 5.
- Se non è possibile unire P e X, allora creare un nuovo neurone nascosto associato a X. Saltare al punto 5.

5. Cancellare i neuroni nascosti che soddisfino la condizione $V_p \leq \beta_v$ o $R_p \leq g_v$. Il pattern X è assegnato alla classe WC.

Processi di base associati alla creazione e aggiornamento dei neuroni nascosti(centri)

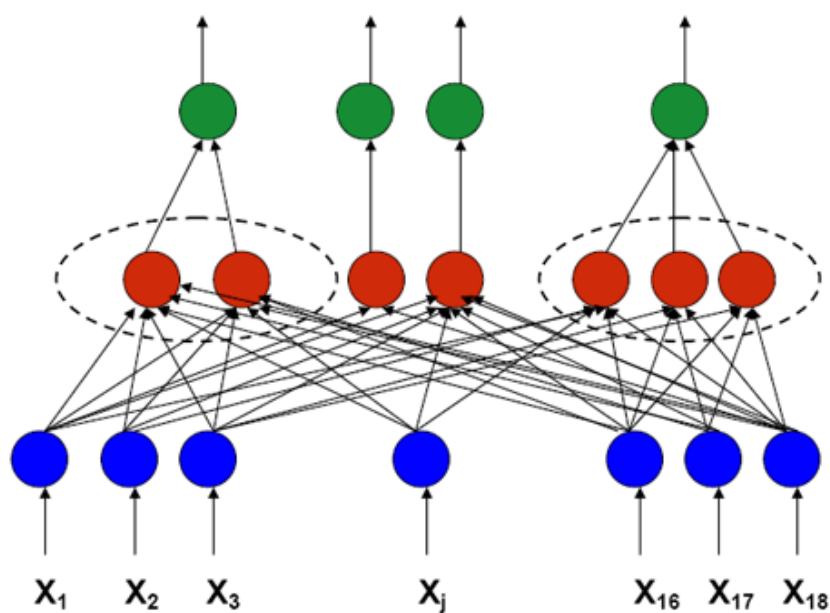
- Creazione di un nuovo neurone nascosto (centro)
- Aggiornamento degli attributi di un neurone esistente, compreso il suo raggio
- Verifica della intersezione delle regioni di classificazione
- Unione di un pattern con un neurone nascosto
- Riduzione del raggio di un neurone nascosto
- Scelta del neurone nascosto vincitore della gara

Il modello neuronale di Sanchez e all. è composto da tre strati di neuroni (vedi figura 3.4). Il numero di neuroni nello strato di ingresso è determinato dalla dimensione dei pattern d'ingresso. Lo strato nascosto contiene i neuroni nascosti o centri che sono stati creati attraverso il processo di evoluzione della rete neuronale, per raggiungere adeguata classificazione dei pattern d'ingresso presentati.

All'inizio lo strato nascosto non contiene nessun neurone e ogni neurone aggiunto rappresenta un vettore caratteristico di un cluster. Lo strato di uscita contiene tanti neuroni quanti classi di patterns. Ogni neurone nello strato nascosto riceve ingressi da ognuno dei neuroni dello strato d'ingresso. Ogni neurone dello strato di uscita riceve ingressi da un sottoinsieme di

neuroni dello strato nascosto.

Figura 3.4: Architettura tipica di una rete dinamicamente evolutiva



Ognuno di questi sottoinsiemi rappresenta un gruppo di vettori caratteristici di una classe e definiscono un cluster. Tutti i neuroni di un cluster si collegano con l'unico neurone che rappresenta la classe e che si trova nello strato di uscita.

La rete neurale costruisce neuroni nascosti (anche chiamati vettori caratteristici o centri) i quali sono combinazione non lineare dei pattern d'ingresso. D'altro canto, ogni neurone ha associato un raggio (o soglia) il quale determina l'insieme di pattern d'ingresso che il neurone può classificare. I neuroni dello strato nascosto si caratterizzano anche per un coefficiente di vitalità. Questo coefficiente rappresenta la probabilità che il neurone si trovi attivo un numero minimo di volte in un numero dato di presentazioni dei pattern d'ingresso.

3.6 Come utilizzare l'approccio delle reti competitive in un problema di localizzazione

Il problema di localizzazione di utenti mobili in un ambiente WLAN può essere visto come un problema di clustering o vector quantization (tutti e due spiegati all'inizio di questo capitolo).

In questo senso, un modello di rete neuronale competitivo ed evolutivo dovrebbe risultare molto adeguato come approccio a questo problema. La proposta si indirizza alla costruzione di una rete neuronale competitiva - supervisionata ed evolutiva per la localizzazione in ambiente WLAN, prendendo come spunto il modello proposto da Sanchez et al..

Capitolo 4

Rete Neuronale: Costruzione mappa radio e localizzazione

L'applicazione definitiva di questa tesi consisterà nello sviluppare un servizio, per ogni utente della facoltà, in grado di fornire la localizzazione di qualunque persona fornita di dispositivo mobile.

Il servizio, che potrebbe essere solamente il primo di una vasta fornitura di servizi per utenti della facoltà, sarà composto da una parte server ed una client. La parte server risiederà in sede amministrativa e sarà quella che si occuperà di gestire la rete neuronale, e modificarla (ad esempio in seguito a variazioni strutturali dell'edificio o eseguendo periodiche ricostruzioni della mappa radio) attraverso nuovi pattern di potenza rilevati nell'ambiente.

Essa dovrà inoltre stare in ascolto delle richieste di servizio dei clients, i quali per effettuare una localizzazione non faranno altro che fornire un pattern di potenza loro per conoscere la propria posizione, o un identificativo di utente per conoscere la posizione altrui.

In questo capitolo, cercando di seguire un percorso incrementale che permetta di assicurare il funzionamento di ogni frazione del progetto si svilupperà la parte server, mondata della gestione del servizio.

Questa parte, che verrà completamente sviluppata e resa funzionante nei capitoli seguenti, consisterà in un' applicazione stand-alone che effettuerà sia gestione della rete neuronale sia localizzazione di pattern di potenza.

Nel capitolo 7 si cercherà di inglobare questo prototipo in un server capace di fornire servizi di localizzazione ai client.

4.1 Analisi del problema

Tornando al percorso che porterà a scegliere localizzazione attraverso reti neurali, sono già stati notati alcuni scenari realizzabili, tutto questo grazie alla possibilità di sfruttare tecniche quali Scene Analysis per confrontare parametri della rete pre-raccolti con quelli dell'utente da localizzare. La somiglianza fra questi parametri permetterà di scoprire dove si trova l'utente.

Questo oggi è possibile anche grazie ai dispositivi sempre più mobili e personali che permettono all'utente di collegarsi in rete senza il bisogno di fili.

Tutto nasce dall'esigenza di localizzare utenti all'interno di un edificio. Le soluzioni possono essere molteplici, si parte però dal fatto che un edificio di questa entità è dotato di una rete standard 802.11b connessa a sua volta con la rete internet.

Si sarebbero potute intraprendere altre strade per localizzare gli utenti, si pensi a telecamere sparse per l'edificio o scansione di iridi o impronte digitali ma anche se ciò porta alla localizzazione fisica della persona e non del suo assistente digitale, con la certezza di localizzare l'utente, questa soluzione risulta maggiormente invasiva sia per l'ambiente che per l'utente.

A questo punto volendo sfruttare dei dispositivi economici ma soprattutto già distribuiti sul territorio, si è percorsa la strada della localizzazione tramite WLAN.

Per fare ciò si dispone di tecniche quali triangolazione dei segnali radio che un dispositivo riceve dai punti di accesso (Access Point - AP) della rete. Queste tecniche però risultano più complesse e poco affidabili in ambienti chiusi.

Per questo si sceglie la tecnica di Scene Analysis. Come già citato questa tecnica si divide in due fasi, quella della costruzione della mappa radio (offline) e quella della localizzazione (online).

Questo progetto si avvale di un attento studio della casistica delle soluzioni comprese le reti neurali. Infatti queste sono portate a risolvere con molta efficienza problemi di clustering e classificazione come quello di associare un utente ad una posizione in base ai parametri della rete WLAN. Avremo due possibilità di risoluzione nella tecnica scelta per la localizzazio-

ne. Ci sarà la soluzione deterministica dove il confronto fra mappa radio e parametro dell'utente sarà fatto unicamente sul valore di potenza o la soluzione probabilistica dove il confronto viene effettuato in base alla densità di probabilità del valore di potenza. Il progetto verterà a fornire una soluzione del primo tipo, lasciando a sviluppi futuri la possibilità di affrontare la soluzione probabilistica.

In primis si riportano le motivazioni per cui la tesi dell'ing. Gramellini si sofferma sulla scelta di utilizzare reti di tipo SOM per eseguire localizzazione di pattern. Queste reti Self Organizing Map erano state individuate proprio perchè adatte a risolvere problemi di clusterizzazione.

"Questa rete combina il principio di apprendimento competitivo con una strutturazione topologica dei neuroni, in modo che i neuroni adiacenti tendano ad avere vettori dei pesi simili.

In una rete auto-organizzativa infatti i neuroni tendono ad avere vettori di pesi che prendono le caratteristiche dello spazio vettoriale di entrata. La relazione di vicinanza si traduce nella prossimità fra i neuroni nello spazio Euclideo.

In queste reti il valore dei pesi vengono modificati attraverso il processo di apprendimento, nel quale quindi ogni vettore dei pesi si muove nella direzione del prototipo di qualche sottoinsieme di pattern di entrata. L'uscita di ciascun neurone dello strato di uscita rappresenta la vicinanza del neurone al pattern di ingresso.

Come conseguenza dell'applicazione dell'algoritmo di apprendimento SOM, i neuroni si organizzano, e quelli vicini, a seconda della topologia, si associano tra essi. In altri termini questi neuroni hanno vettori dei pesi con valori vicini fra loro a seconda dello spazio di entrata." [1]

La rete dovrà implementare la tecnica di Scene Analysis perciò dovrà supportare due fasi, quella offline e quella online. La prima potrebbe essere vista per la rete come una fase di addestramento, mentre la seconda una fase di funzionamento effettivo.

Il problema scaturisce dal fatto che le reti SOM sono reti con una struttura già ben definita alla creazione, mentre la fase offline parte da zero con l'inserzione di array di potenze rilevati nell'ambiente. Questo ha spinto il

progetto nell'intraprendere lo sviluppo di una rete evolutiva, come quella definita da Sanchez et al. [8]

La rete evolutiva, che perciò dovrà sostenere una fase offline, sarà sottoposta ad un addestramento che dovrà essere "guidato", o meglio supervisionato. In materia di apprendimento distinguiamo le reti in due grandi categorie.

- Reti con apprendimento NON supervisionato
- Reti con apprendimento supervisionato

Le prime avranno come input pattern casuali, non guidati, che permetterebbero alla rete di evolvere autonomamente creando una clusterizzazione dettata solamente dai propri parametri.

Le seconde, molto più interessanti in un problema di clusterizzazione, accetteranno come input durante la fase addestrativa pattern con già definito il cluster d'appartenenza. In questo modo la classificazione che la rete effettuerà sarà guidata, al fine di costruire una mappa interna che consideri ogni neurone facente pattern di un ben determinato cluster.

Ovviamente in un problema di localizzazione i cluster saranno le stanze di un edificio e i pattern per addestrare la rete avranno con se l'identificativo della stanza dalla quale sono stati raccolti.

In questo modo i pattern andranno a formare una costellazione di neuroni nascosti suddivisi in cluster, ai quali ogni pattern avrà fatto riferimento nella fase d'addestramento.

4.1.1 Struttura

Si necessita il bisogno di spezzare lo sviluppo della rete in due parti.

La prima è quella di realizzazione della rete stessa, cercando però ad ogni modo di avere un sistema capace di costruire qualsiasi tipo di rete con la propria architettura, le proprie regole d'apprendimento, d'attivazione e di propagazione.

Non si vuole che la rete utilizzata in questo progetto diventi vincolante, al fine di poter riutilizzare la struttura per un altro sistema che impieghi una diversa rete neuronale artificiale.

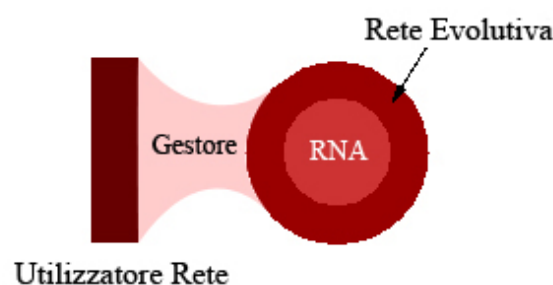
Si desidera perciò costruire una rete generica che permetta di essere specializzata in qualsiasi altra rete. Nel nostro caso la specializzazione sarà quella di una rete evolutiva.

Si mirerà innanzitutto ad ottenere tale rete, ma ancor di più un'applicazione a lunga visione che permetta con il minimo sforzo di ottenere qualsiasi tipo di rete. Infatti, immaginando che nel breve periodo venga realizzata anche una soluzione probabilistica del problema di localizzazione, si denoterà il necessario utilizzo di una rete diversa.

L'ideale sarebbe dividere in due strati (a "cipolla") il sistema, ove quello interno sarà il cuore della rete non specializzata e uno strato esterno che implementa le caratteristiche della rete evolutiva. Questa cipolla verrà poi utilizzata dall'applicazione in oggetto sviluppata nella seconda fase.

La seconda fase consiste nello sviluppo del gestore della rete. Esso dovrà fare da intermediario tra la rete, fornendone i servizi, e chi la vuole utilizzare. Si potrebbe pensare ad un controllore generico inserito in ogni rete ma ai fini del sistema risulta più sensato riporlo a lato applicativo, mantenendo il suo onere quello di comunicare con gli altri componenti MVC (fig 4.1).

Figura 4.1: Struttura macrocomponenti del sistema



Struttura di una rete neurale generica

Ci si avvale per questo di una tesi che ha difatti già affrontato il problema [9]; lo schema principale sta nell'analizzare due sottosistemi che finiranno per comprendersi, il primo quello riguardante la tassonomia della rete e il secondo inerente la tassonomia dei neuroni.

Sappiamo che in una rete neuronale ci possono essere tre tipi di neuroni che si differenziano dalla tipologia di strato a cui essi appartengono. Semplificando il set di tipologie distinguiamo principalmente tre strati. Quello di ingresso, quello nascosto (intermedio) e quello di uscita.

Ogni neurone ha delle caratteristiche comuni ma alcune differenze che nascono dallo strato di appartenenza. Nella tesi sopraccitata si definivano tre tipologie di neuroni a seconda del loro strato di appartenenza. Per questa tesi non sarà così.

Infatti si opterà per la scelta di accumulare tutti i neuroni in un'unica specializzazione, quella del neurone stesso. Si fornisce alla rete la possibilità di trattarli in maniera diversa a seconda dello strato di loro appartenenza che sarà quindi un'informazione della rete e non più del neurone. In questo modo il neurone diventa un ente un po' più "stupido" ma svincolato dallo strato di appartenenza. Sarebbe ridondante che ogni neurone portasse con sé le informazioni sulle caratteristiche di strato, uguali per ogni neurone di uno stesso strato. Al bisogno la rete fornirà al neurone le caratteristiche che gli necessitano per svolgere il compito del proprio strato. Inoltre, se un domani le tipologie di strato, che una rete gestisce, aumentassero, non si dovrà ridefinire una tassonomia di neuroni. Il problema di una tassonomia predefinita nasce quando si pensa ad una rete neuronale completamente connessa. Infatti questa dovrà permettere ad ogni neurone di essere collegato in ingresso o in uscita a qualsiasi altro, di qualsiasi strato.

Si potrebbe pensare di associare ad ogni neurone, potenzialmente collegato ad un altro, un peso che potrebbe avere un valore nullo in assenza del collegamento. Ma come differenziare un peso dall'altro? Si potrebbe costruire una matrice dei pesi per ogni neurone ove le righe rappresentino ognuna lo strato e le colonne il neurone dello strato. Si pensi alla comodità di applicare una funzione random per generare in principio una pesistica casuale, procedimento svolto nella maggior parte di casi per inizializzare reti. I neuroni dello strato d'ingresso non hanno pesi. Ma si pensi a reti come quella backpropagation ricorrente: essa infatti avrà neuroni che messi allo strato d'ingresso prenderanno input dai neuroni degli strati successivi. La scelta di non specializzare i neuroni è diventata una sicurezza.

I neuroni oltre ai pesi dovranno possedere anche dei valori di uscita che rappresentano il potenziale del neurone stesso, che sarà l'ingresso per altri

neuroni.

In seguito si definisce un componente che rappresenti la rete. Questo componente dovrà fornire funzioni agli enti esterni per agire su di se, modificare l'architettura e i parametri che lo definiscono. La rete neurale è caratterizzata da tre caratteristiche principali per ogni strato:

- Architettura
- Funzioni d'attivazione
- Funzioni di propagazione

La rete deve poter fornire gli strumenti per modificare la propria architettura. Essa conterrà un vettore di neuroni per ogni strato che ha al suo interno.

La rete dovrà inoltre definire i parametri inerenti ad ogni strato.

Un altro importante punto sarà come la rete dovrà attendersi il pattern d'ingresso. Questi infatti dovranno poter essere inseriti in qualsiasi lunghezza e tipologia. Ad esempio per il sistema di localizzazione che si intende sviluppare il pattern sarà di lunghezza pari agli access point utilizzati e con valori numerici rappresentanti le potenze registrate nei singoli punti campionati.

Ulteriore aspetto sarà differenziare i pattern di addestramento da quelli di funzionamento "online" della rete. Per questo si può ricorrere nel differenziare i pattern inseriti. Si potrebbe avere una specifica entità che rappresenta il pattern da dare in input alla rete. Si potrebbe anche fornire in due modi diversi alla rete dei pattern, uno per addestrarla, e uno per utilizzarla. Ad esempio, i patterns per una rete supervisionata potrebbero già contenere al proprio interno il cluster di appartenenza.

Ad una rete bisogna inoltre permettere di salvare il proprio stato e la propria struttura su database o file di testo per poter tornare in quello stato in un secondo momento. Quindi servirà anche una funzionalità che permetta di leggere lo stesso file o database per avviare la rete da uno stato predefinito.

Una rete dovrà permettere anche di fornire l'uscita in seguito alla rice-

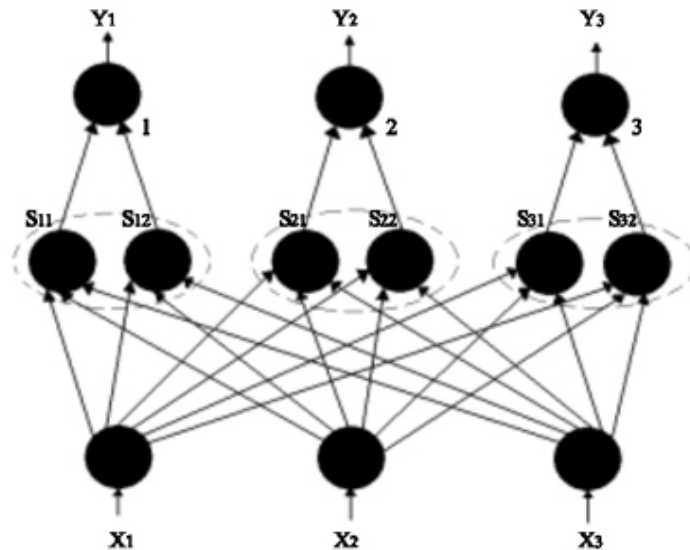
zione di un pattern. Nel caso in questione, servirà quindi una funzione che permetta di ottenere il cluster inerente al pattern inserito.

Struttura di una rete neuronale evolutiva

La caratteristica di una rete specializzata come quella che si vuole implementare è che la struttura non è fissabile ma cambia nella fase di addestramento. Infatti i neuroni iniziali sono solo quelli dello strato d'ingresso, pari al numero di access point che potrebbe anche variare nel tempo.

Avremo nella rete parametri quali il raggio, la dimensione del cluster, il fattore di modifica del raggio, la soglia per definire se scartare clusters a valori troppo bassi. Tutti questi parametri saranno implementati in una rete ad hoc, che specializza la rete di base descritta nella sezione precedente. Ovviamente i neuroni avranno matrici di pesi ridotte, perchè si ricorda che i collegamenti di una rete di questa topologia sono quelli riportati in fig 4.2.

Figura 4.2: Struttura rete Evolutiva



Per ogni strato verranno definite le funzioni di attivazione e propagazione, oltre a quella che valuterà la distanza euclidea fra il vettore dei pesi di un neurone e il pattern (la distanza euclidea potrebbe venire in futuro sostituita da altri tipi di distanza). Una funzione di addestramento permetterà alla rete di evolvere grazie ai pattern inseriti in ingresso e un'ultima funzio-

ne eseguirà la localizzazione vera e propria. La rete potrà quindi fornire in qualsiasi istante l'uscita derivante dall'ultimo pattern inserito.

La struttura dovrà essere molto dinamica e conferire la possibilità alla rete di modificarsi anche una volta finita la fase di addestramento, perché l'edificio non rimarrà immutato per sempre.

Struttura del sistema che usa la rete

Questo sistema dovrà permettere l'utente di interagire con la rete per addestrarla e localizzare pattern. Infatti in questo primo prototipo il sistema sarà unico, quindi pure l'interfaccia è unica, sia per chi amministra la rete, sia per chi vuole localizzare utenti.

Seguendo il pattern Model, View e Control, la rete rappresenterà la parte Control, il View l'interfaccia e il Model i dati immagazzinati inerenti alla rete. Di seguito si riporta uno schema che aiuti a farsi un'idea dell'architettura che sta nascendo. E' chiaro che si deve pensare ad un sistema distribuito, che possa gestire la concorrenza di più utenti che facciano richiesta di servizio; allo stesso tempo si dovrà permettere l'amministratore di mettere in stand by la rete dall'utenza per l'aggiornamento. Questa rete deve perciò vivere sempre, con la possibilità di essere fermata e riavviata per fasi diverse (offline, online).

4.1.2 Comportamento

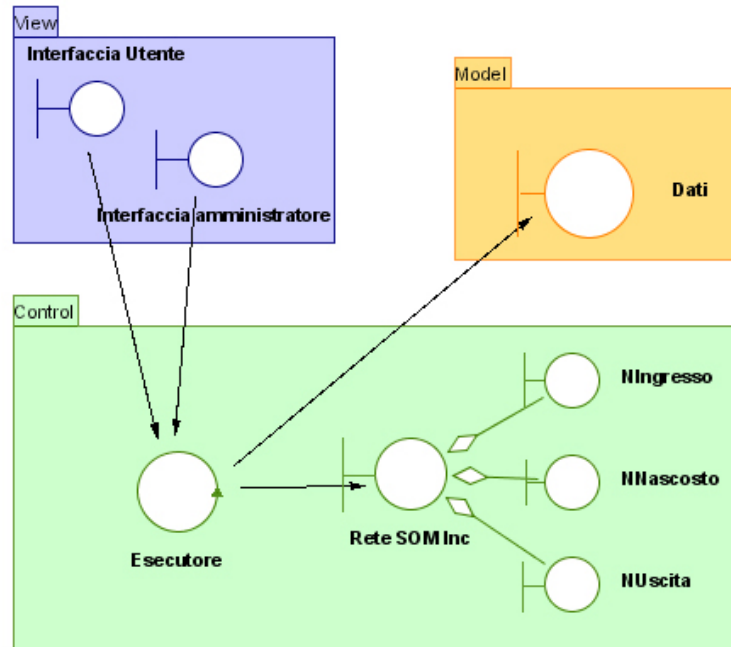
Comportamento di una rete Evolutiva

Una rete di questa entità comunica all'esterno solo attraverso le uscite e ha come input gli unici pattern di addestramento e localizzazione che avranno dimensione variabile, solitamente con il numero di access point.

Ci saranno dei parametri per variare il comportamento della rete in questione. Uno sarà il raggio, che definirà quanto far distare al minimo due neuroni dello strato nascosto diversi, altri riguarderanno l'eventuale spostamento dei neuroni in seguito ad addestramento.

Permettere all'amministratore di far evolvere la rete come rappresentazione del mondo esterno (struttura dell'edificio o sua capienza) necessiterà di numerose considerazioni. Trattandosi di una rete supervisionata non basterà

Figura 4.3: Struttura proposta per l'applicazione



fornire come input i pattern di addestramento ma servirà anche comunicare a che cluster appartengono.

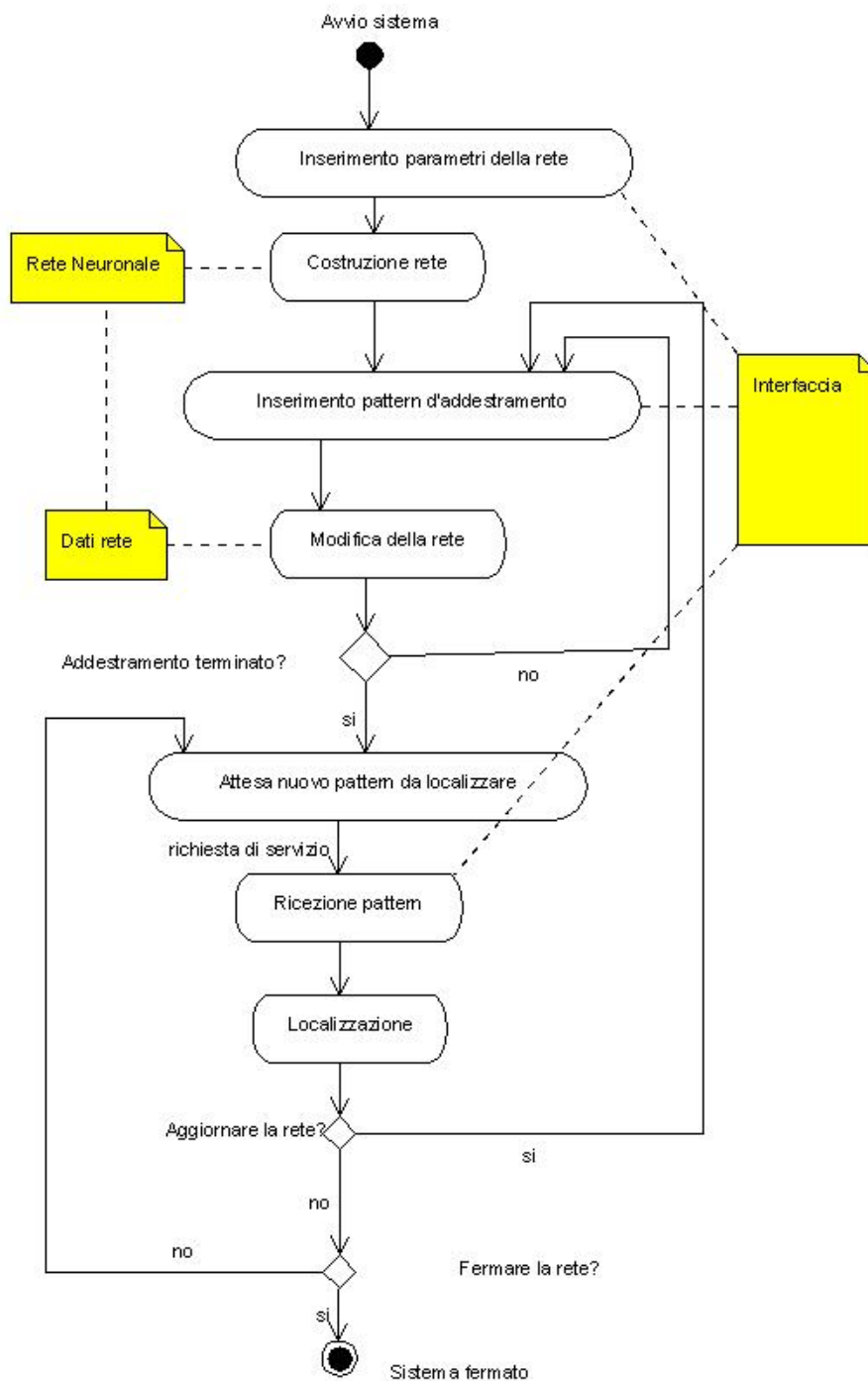
Una volta inseriti tutti i pattern d'addestramento inizierà la fase online, ove la rete otterrà, dai richiedenti del servizio, dei pattern di potenza misurati e risponderà con il corrispondente cluster, ovvero la localizzazione.

Comportamento del sistema nella fase offline

Il sistema effettivamente una volta avviato dovrà subito affrontare questa fase, che per semplicità, in questa versione del sistema, non si ripresenterà (in un sistema reale si pensi ad un costante aggiornamento della rete a causa di mutamenti ambientali).

Un'interfaccia pilotata dall'amministratore dovrà comunicare il tipo di rete e i parametri che la riguardano. In seguito una numerosa flotta di pattern d'addestramento faranno sì che la rete cresca e si modifichi. Serve che la rete assuma i pattern differenziati per cluster reali (informazione che l'utente dovrà inserire) perciò già conoscendo questi bisognerà associare ad ogni pattern il valore del cluster di appartenenza.

Figura 4.4: Comportamento dell'applicazione



Si tenga conto che l'addestramento può riprendere continuamente nel corso di vita della rete.

Comportamento sistema fase online

Finita la precedente fase il servizio diventa accessibile agli utenti. A questo punto bisognerà gestire una coda di richieste per evitare problemi di collisione. Ad ogni richiesta, fatta attraverso la fornitura di un pattern da riconoscere, seguirà una risposta, elaborando le uscite dei neuroni dell'ultimo strato della rete, che sarà la localizzazione del pattern.

In seguito si potranno fornire upgrade al sistema per fare aggiornamenti, manuale o perfino automatico, della struttura della rete, in seguito a mutamenti dell'ambiente o difficili riconoscimenti di pattern.

4.1.3 Interazioni

Figura 4.5: Tabella delle interazioni

Entità emittente	Entità ricevente	Segnale di interazione	Aspettativa dell'emittente	Azione del ricevente
Interfaccia amministratore	Gestore rete	Parametri rete	Ottenere una rappresentazione della rete	Inizializzazione rete
Interfaccia amministratore	Gestore rete	Inserimento pattern d'addestramento	Riconoscere l'incrementare della rete	Rete che impara e si modifica
Interfaccia utente	Gestore rete	Richiedere localizzazione	Ottenere posizione	Processazione del pattern del richiedente
Gestore rete	Gestore dati	Salva stato	Lo stato della rete potrà essere ristabilito	Salvataggio dello stato della rete
Gestore rete	Gestore dati	Ripristina stato	Ripristino di un certo stato della rete	Recupera i dati relativi ad uno stato della rete

Considerazioni

Dalle interazioni prese in considerazione risulterà importante definire un comportamento fra i vari attori del sistema. Una funzione fondamentale sarà svolta dal gestore della rete, colui che consentirà di interagire con la rete e salvarne lo stato. Siccome questa parte del progetto è rivolta a risolvere il

problema della rete, ovvero il cuore del sistema, ci si occuperà con minore interesse del dialogo fra i diversi componenti del sistema, cosa che verrà affrontata in un secondo momento.

4.2 Progetto

Il sistema verrà sviluppato seguendo le indicazioni già date in analisi con un linguaggio ad oggetti, in questo caso si preferisce il linguaggio Java per la sua compatibilità con qualsiasi piattaforma oltre alla sua notevole diffusione.

La rete neurale supervisionata

Per la costruzione della rete, come già citato, si partirà dallo sviluppo di una rete generica, che potrà essere specializzata per il compito specifico che essa dovrà svolgere. La rete generica sarà composta da tre elementi principali, uno rappresentante i neuroni, uno la rete e infine i pattern da inserire nella rete, intendendo questi come fattori comuni ad ogni tipo di rete.

Nella specializzazione del caso, la rete verrà sviluppata separando le due fasi tipiche di Scene Analysis, quella offline e quella online.

La prima (offline) determinerà la costruzione della mappa radio, della quale si farà utilizzo nella fase successiva.

La seconda fase (online) utilizzerà la rete addestrata (mappa radio) per fare localizzazione dei pattern. Si dovranno stabilire le funzioni di attivazione e propagazione per ogni strato.

Il sistema che usa la rete

Dopo la costruzione di una rete specifica funzionante, il progetto verterà sullo sviluppo del sistema che usa la rete neurale, tramite il modello MVC già visto in analisi.

Si avrà quindi un esecutore di comandi sulla rete, quali la creazione, l'addestramento, la localizzazione. Esso dovrà anche gestire il salvataggio della rete. Per permettere ciò si dovrà utilizzare una seconda rete che renda pubblici i dati della prima.

Infatti nella prima rete i dati rimarranno protetti. Si pensa perciò ad una rete ancora più specializzata da utilizzare per recuperare informazioni.

Quindi i componenti che si vanno a definire sono una rete Neuronale Evolutiva più pubblica, un gestore di essa e una console di comando per essa.

L'idea di dividere questi tre macro componenti nasce per lasciare l'opportunità di sostituire uno dei componenti in futuro, al fine di migliorarlo senza dover ricostruire da capo il sistema.

Essendo una rete supervisionata, come pattern si pensi ad uno intelligente che abbia già con se il cluster di appartenenza per l'addestramento.

4.2.1 Rete

Rete Generica

Il componente principale è la rete, questo permetterà di definire la struttura e le parti comuni alle reti neurali di qualsiasi tipo. Si dovrà però fare una ulteriore scelta fra il tipo di reti che si vogliono prediligere. Infatti si differenziano le reti che hanno una struttura definita in fase di inizializzazione da quelle evolutive.

Queste ultime infatti avranno una fase di costruzione dinamica, che dovrà permettere ai neuroni di essere gestiti con facilità anche nella loro struttura. Come già ribadito in analisi la rete del sistema, che specializzerà in questo caso quella generica, sarà del tipo incrementale ovvero attraverso la fase di offline (addestramento) essa otterrà il suo aspetto definitivo per compiere il proprio mestiere. Una specializzazione di questa rete generica potrà bastare per permettere a reti di altri tipi di essere implementate.

Se la rete da utilizzare avrà già una struttura predefinita non verrà eseguito alcun addestramento. Se invece, come in questo caso, la struttura finale è opera di un addestramento incrementale bisognerà definire lo svolgimento di quest'ultimo. Infatti esso oltre a variare i parametri varierà anche l'architettura della rete. In seguito si noteranno quali sono le caratteristiche che i neuroni dovranno possedere per sostenere una simile dinamicità.

La rete dovrà possedere strumenti per l'aggiunta di neuroni e la loro gestione. Si pensi ad una struttura ad array, dove ogni strato della rete è composto da un array di neuroni. Questa struttura è compatibile con tutte le reti neurali. Infatti sarà possibile definire alla costruzione della rete il numero di livelli e il numero di neuroni in ognuno di essi.

Si predispone quindi di un metodo per addestrare la rete, questo avviene attraverso l'inserzione di pattern (componente da definire), in questo modo la rete dovrà sollecitare la modifica dei propri neuroni e degli stessi strati che li contengono.

I neuroni saranno elementi piuttosto ingenui, come vedremo in seguito, si pensa perciò di definire le loro funzioni a livello di strato. Sarà la rete che gestirà il loro comportamento a seconda dello strato di appartenenza. Questo tornerà utile in un eventuale cambiamento del loro comportamento, basterà modificare la rete per avere neuroni con funzioni diverse.

Si necessita quindi di funzioni di strato. Per far sì che i neuroni eseguano le proprie funzioni si metteranno a disposizione metodi che contraddistinguono il tipo di funzione e di strato; questi saranno richiamabili dai singoli neuroni per effettuare le sinapsi.

La rete metterà a disposizione del sistema un metodo per permettere l'esecuzione delle funzioni dei neuroni di ogni strato. Di questo se ne parlerà in seguito nella sez 4.2.2.

Oltre ad un metodo per l'addestramento (training), nel quale l'architettura verrà costruita o modificata, si necessita di un metodo per far funzionare la rete (doJob). Quest'ultimo non restituirà alcun valore ma permetterà alla rete di modificare i propri valori per giungere al target di essa. Starà poi alla specializzazione scegliere come farlo. Solitamente ciò consisterà nel mettere in esecuzione le funzioni di propagazione e attivazione dei vari strati della rete.

Si avrà una classe astratta che implementerà l'interfaccia di fig 4.7, al fine di permettere alle classi specializzate di definire i suoi metodi, come ad esempio le funzioni di strato.

In seguito la rete verrà arricchita da un identificativo per la possibilità di avere più reti neurali che fanno localizzazione in uno stesso sistema, magari per ambienti diversi.

Figura 4.6: Iniezione delle funzioni

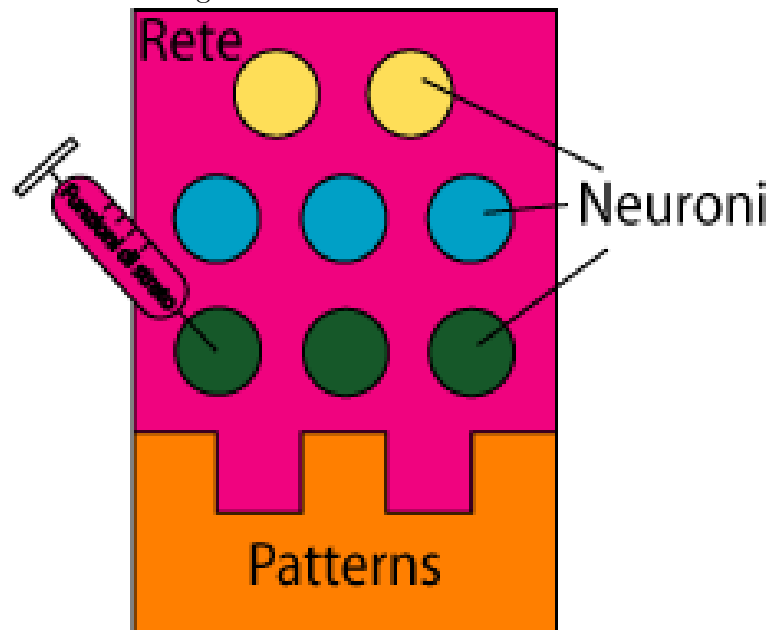


Figura 4.7: UML dell'Interfaccia della Rete

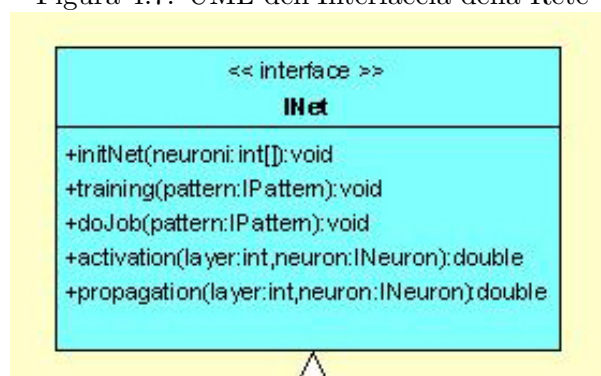
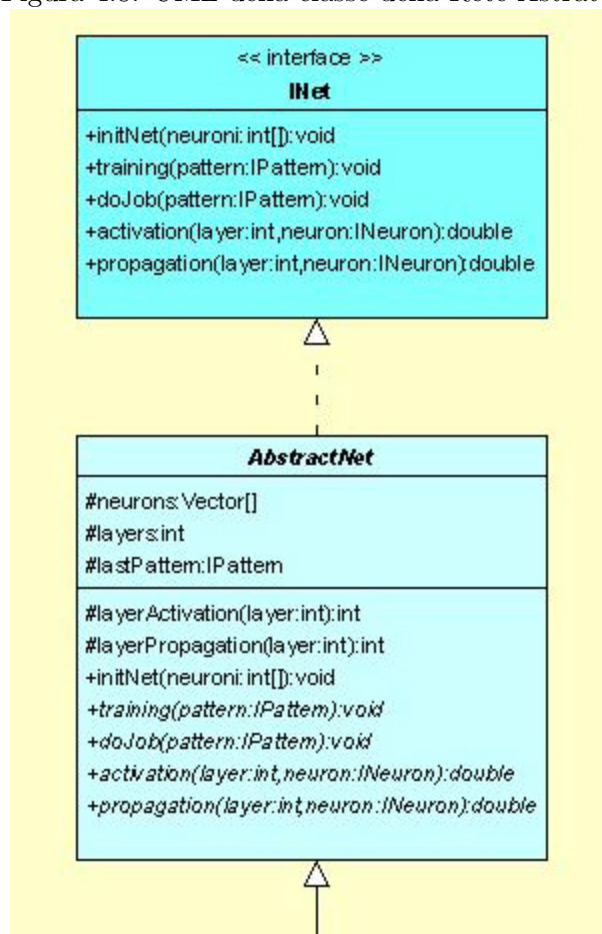


Figura 4.8: UML della classe della Rete Astratta



Rete Evolutiva: FASE OFFLINE

La rete evolutiva estenderà quella astratta ed implementerà quelle parti inerenti all'addestramento, alla localizzazione e alle funzioni di strato. Queste infatti sono le caratteristiche che maggiormente differenziano rete da rete.

Si ricordi che l'architettura per questo tipo di rete cresce dinamicamente nella fase di addestramento. Un'altra rete dinamica varierebbe questa fase con un diverso algoritmo, mentre una rete non dinamica fisserebbe questa fase con una costruzione statica della struttura.

Sicuramente si dovranno conoscere oltre al numero di neuroni in ingresso (numero access point), anche il numero di neuroni di uscita (numero cluster individuati a priori nell'ambiente). Perciò oltre all'algoritmo di addestramento si potrebbe specializzare il costruttore della rete in modo da permettere l'inserzione del numero di neuroni d'uscita nella fase d'inizializzazione. Si potrebbe però fare affidamento ad un'altra soluzione. Infatti si potrebbero creare al bisogno neuroni d'uscita a seconda dei cluster di appartenenza dei pattern d'addestramento senza preimpostarlo alla costruzione della rete. Questa soluzione verrà preferita.

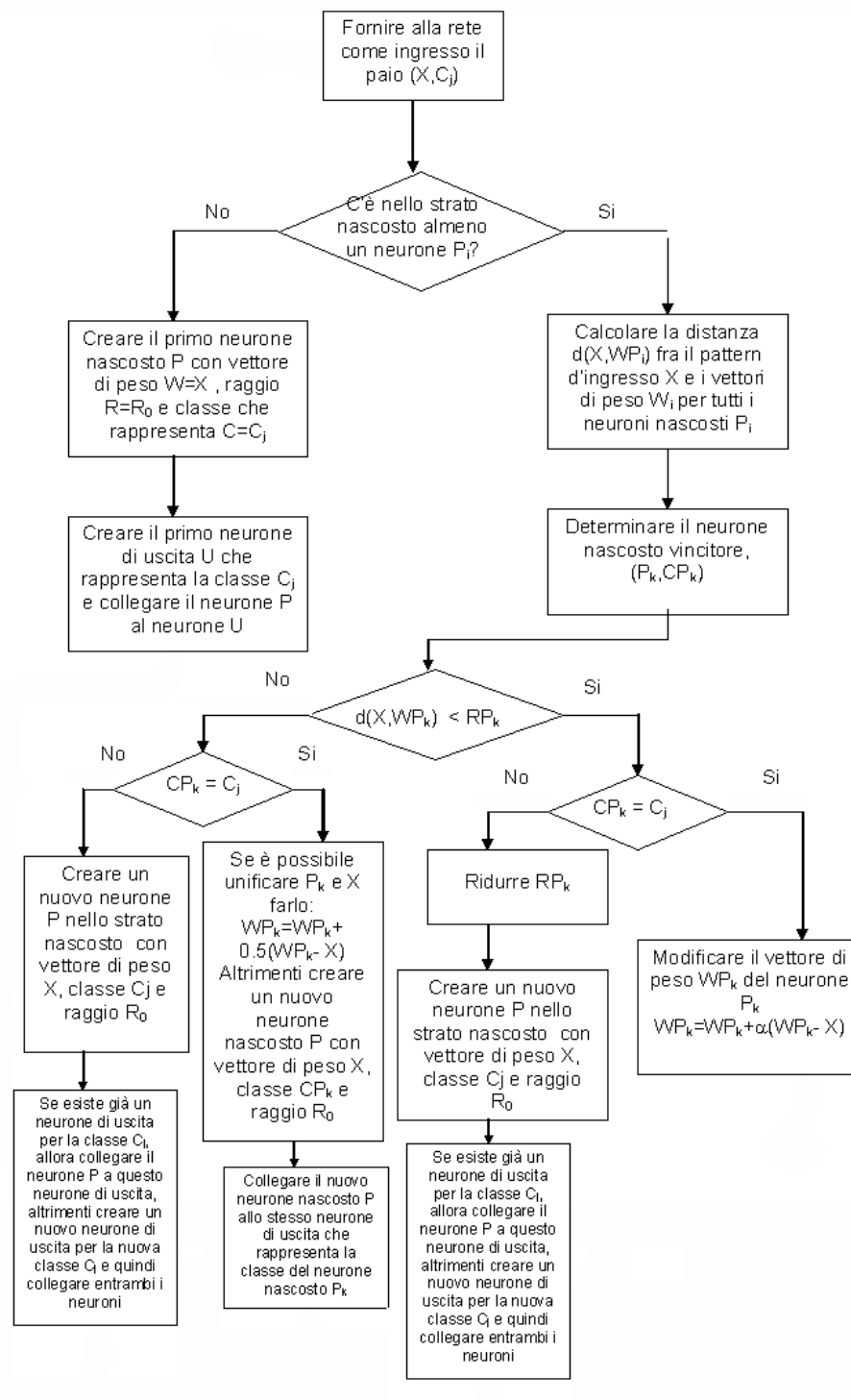
Di seguito si riporta l'algoritmo di addestramento per un pattern, in seguito si analizzeranno le sue caratteristiche che guideranno la costruzione della rete.

Ogni pattern d'ingresso sarà accompagnato da un classe (cluster) d'appartenenza. Questo porterà ad avere dei pattern d'addestramento con più informazioni.

Ad ogni addestramento la rete terrà conto della classe a cui appartiene il pattern, per poter mantenere una statistica dei pattern inseriti per l'addestramento della rete e del cluster di loro appartenenza. In realtà si potrebbe lasciare la gestione di queste informazioni al gestore della rete ma siccome potrebbe accadere che un gestore venga sostituito con un altro (nell'ottica dove ogni componente è un'entità aggiornabile), si preferisce rimanga compito della rete mantenere la gestione dei dati a lei riguardanti, in modo che la sostituzione del gestore non comprometta questi dati.

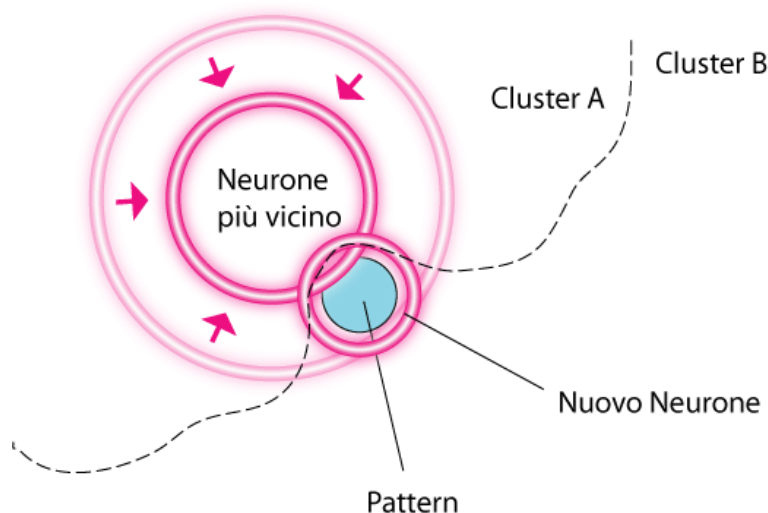
I neuroni non avranno solo un raggio iniziale (R_0) ma anche un massimo

Figura 4.9: Algoritmo d'addestramento della rete Supervisionata



(R_{max}) e un minimo (R_{min}) per questo. Le motivazioni sono date dal fatto che non si potrà estendere a dismisura o ridurre troppo il raggio di un neurone per non inglobare pattern che non gli appartengono. Infatti nel caso in cui un pattern (X di cluster C_j) sia all'interno del raggio di un neurone ma non appartiene allo stesso cluster (CP_k) significa che il raggio del neurone va rimpicciolito per far spazio ad un nuovo neurone che rappresenti il pattern. Il rimpicciolimento deve essere limitato per non far scomparire il neurone.

Figura 4.10: Rimpicciolimento del raggio di un neurone

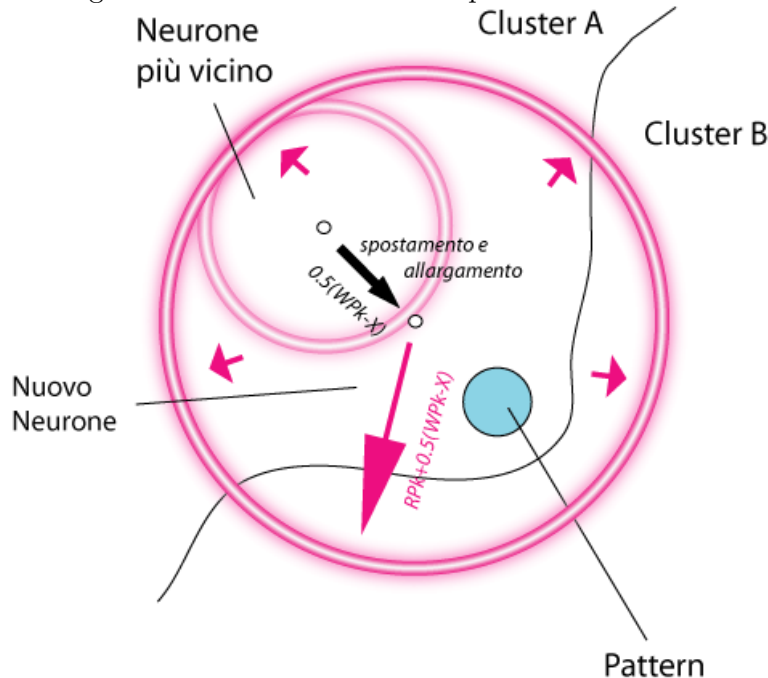


Nel caso invece che un pattern (con cluster C_j) sia fuori dal raggio del neurone più vicino, ma faccia parte del cluster (CP_k) bisogna che il neurone vicino lo inglobi allargando il suo raggio. Ma ciò non potrà essere fatto a dismisura per limitare l'intersecamento delle aree di ciascun neurone. Dovremo considerare anche lo spostamento.

Ridurre il raggio, nel caso di pattern nuovo all'interno del raggio di un neurone di una classe diversa, significherà ridurlo tanto quanto basta per escludere dal vecchio neurone il pattern che ne formerà uno nuovo nascosto.

Quest'ultimo caso è detto "unificazione" ovvero quando un pattern viene inglobato all'interno di un neurone che aumenterà di dimensione. Ci sarà in concomitanza anche lo spostamento del neurone. Questo avviene se il pattern non si trova all'interno del neurone a lui più vicino ma fa riferimento allo stesso cluster. L'unificazione potrebbe non avvenire se è vera una delle seguenti affermazioni:

Figura 4.11: Unificazione di un pattern col neurone



- La distanza fra neurone e pattern è maggiore del raggio max (non può allargarsi tanto da inglobare il pattern)
- L'unificazione altera la classificazione nel neurone (non vengono più coperti i pattern precedenti)
- Il raggio non può più crescere

Si potrebbe riassumere il caso unificazione dicendo che essa può avvenire solamente se spostando e allargando il raggio del neurone per permettere di coprire sia il nuovo pattern che i vecchi non si superi il raggio max. Spostando il neurone di metà della distanza fra esso e il pattern, se il raggio aumenterà solo della distanza per includere il pattern nuovo sicuramente escludo i vecchi pattern. Allora si dovrà sempre allargarlo sino a continuare ad includere i vecchi pattern, cioè ampliarlo della distanza fra vecchia e nuova posizione, ovvero la metà della distanza euclidea calcolata in principio fra pattern e neurone (vedi fig 4.11).

E' importante che ogni modifica del raggio non permetta il superamento dei limiti di minimo e massimo fissati. Nel qual caso si fisserà il valore limite

al raggio.

Il parametro di learning rate (α) è importante per definire di quanto spostare i neuroni a cui vengono affidati nuovi pattern di addestramento entro il loro raggio d'azione.

Si avrà un valore soglia per escludere neuroni dell'ultimo strato con un valore di uscita talmente basso da non permettersi di partecipare alla gara di localizzazione.

Sarà da tener conto lo spostamento del neurone. Questo, che consisterà nella variazione dei pesi del neurone nascosto verrà effettuato modificando tutti i pesi (dimensioni) della differenza fra vecchio valore e valore del pattern corrispondente oltre ad eventuali parametri correttivi (es: learning rate).

Ad ogni inserzione di pattern d'addestramento, esso avrà con sé l'identificativo del suo cluster. La rete farà in modo di associare ad esso il neurone d'uscita che porta lo stesso identificativo. Se nessun neurone è già presente con quell'identità verrà costruito. Tale identificativo che verrà associato al neurone d'uscita servirà anche ad identificare quale cluster ha vinto una gara di localizzazione.

Si vuole tenere in considerazione un ulteriore dato all'interno della rete. Infatti una rete all'attuale stato dell'arte non memorizza dati per identificare gli access point dai quali si riceve potenza.

I neuroni d'ingresso non hanno informazione riguardo essi e l'inserimento di valori di potenza attraverso pattern non ha alcun ordine. In seguito ad un addestramento la rete riceve pattern con valori di potenza in una certa sequenza che potrebbe non essere quella utilizzata per un ulteriore addestramento o localizzazione.

Si necessita dunque di informazioni aggiuntive per i neuroni d'ingresso, queste saranno i MAC degli access point che loro rappresentano. Potrà venir fatto affiancando un vettore per raccogliere codeste informazioni. Tenendo a mente che la costruzione dei neuroni d'ingresso avviene una sola volta, ovvero

alla creazione della rete, questi dati dovranno essere inseriti manualmente.

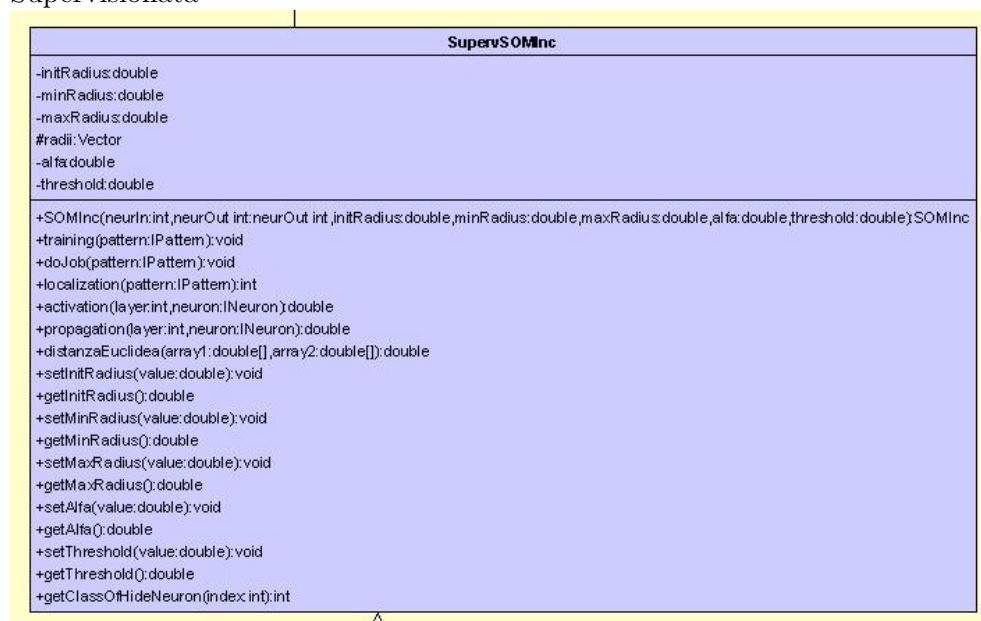
Essi dovranno poter essere salvati su file con la rete per non essere persi quando essa viene aggiornata.

Inoltre ad ogni addestramento da file deve avvenire un controllo per non incappare nell'errore di inserire potenze in un ordine scorretto. La rete sino ad ora costruita non è in grado di aumentare il proprio numero di neuroni d'ingresso fissati in costruzione, perciò pattern di addestramento con valori da access point non registrati non dovranno essere considerati.

Conviene citare una modifica alla rete, in seguito alla considerazione della possibilità di avere pattern di addestramento menomati di qualche componente. In questo caso se l'addestramento comporta lo spostamento del neurone non verrà effettuato sulle componenti mancanti, mentre la creazione di uno nuovo porterà i suoi pesi ad assumere un valore predefinito (molto basso) per le componenti mancanti.

La fig 4.12 mostra in formato UML la classe che rappresenta la rete evolutiva.

Figura 4.12: UML della classe che rappresenta la Rete Evolutiva Supervisionata



In seguito all'inserimento di un ulteriore campo nei pattern, ovvero quello dell'identificatore di ogni componente, che nel progetto risulterà il MAC

dell'AP, la rete è stata fornita di un sistema di ordinamento automatico delle componenti. Infatti via via che il progetto procederà, si abbandonerà la scelta di addestramento manuale, lasciando solo quello da file, output dell'applicazione per rilevare le potenze. Grazie all'addestramento da file si potrà sempre conoscere l'identità della componente del pattern, per questo nel caso in cui un pattern venga inserito con una componente che la rete non riconosce o mancante, non si farà altro che assegnare a quelle mancanti un valore prefissato, teoricamente zero Watt (in realtà -100 dBm).

La rete inoltre si è avvalsa nel cammino della sua costruzione di identificatori per ogni cluster, così da non essere vincolati da valori numerici che rappresentavano in un ambiente stanze e studi. Grazie a identificatori stringa è possibile utilizzare ancora l'output dell'applicazione di rilevazioni potenze per ricevere pattern d'addestramento appartenenti a cluster nominati dall'utente all'atto delle misurazioni.

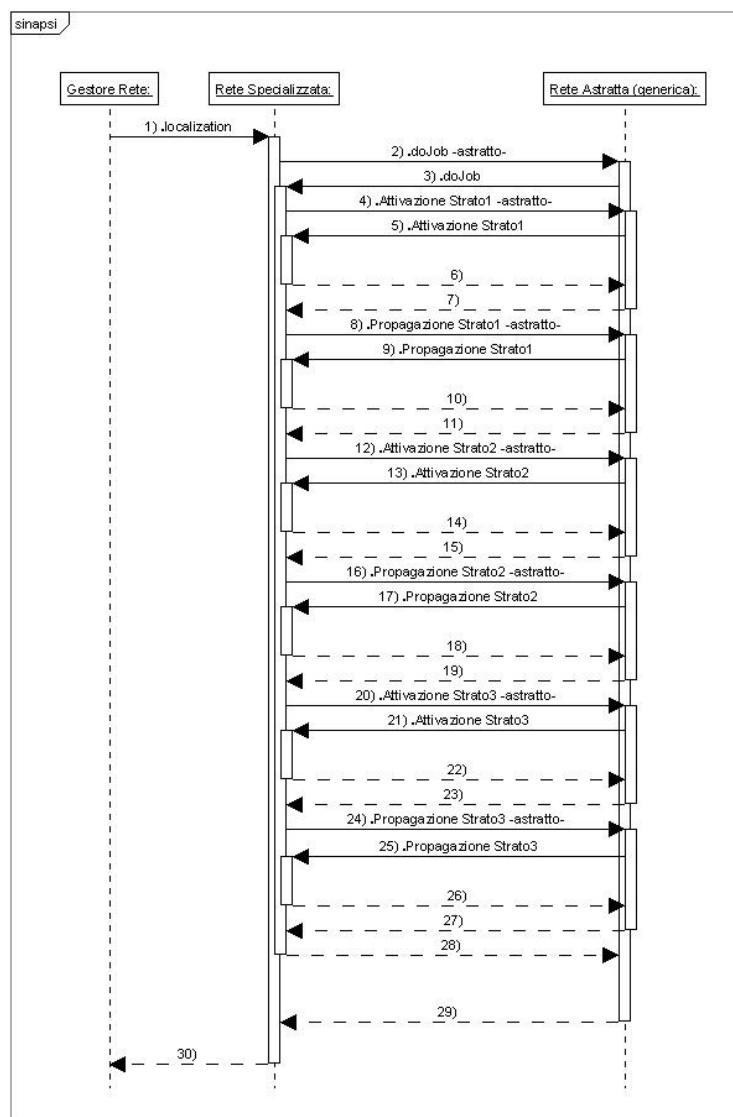
Rete Evolutiva: FASE ONLINE

Come già citato avremo due funzioni di rete per ogni strato, ingresso, nascosto e uscita, che saranno gli unici presenti in questo progetto. Quando la localizzazione avrà luogo ogni neurone verrà richiamato ad eseguire le proprie operazioni definite nella rete. Il metodo (localization) che eseguirà questo procedimento sarà tipico di questa rete, e non farà altro che richiamare la funzione standard di funzionamento rete (doJob), e prelevare dalla rete i risultati che interessano (uscite dell' ultimo strato).

Pure il metodo standard di funzionamento rete sarà implementato nella specializzazione perchè cablarlo nel generale non permetterebbe alle reti di fare eventuali elaborazioni fra calcolo di funzioni fra uno strato e l'altro. Si ricorda che le funzioni di attivazione e propagazione di uno strato sono da implementare nella rete specializzata ma in automatico vengono eseguite da quella generica una volta invocato il metodo doJob della rete. Con l'esecuzione delle funzioni di attivazione e propagazione della rete avvengono le sinapsi fra i neuroni di strati adiacenti. (fig 4.13).

Si passa ora alle funzioni della rete in questione. Essendo lo strato d'ingresso creato nella fase d'inizializzazione rete, il pattern d'ingresso dovrà essere di una lunghezza predefinita (bisognerà controllare che non ci sia stato un errore nel pattern).

Figura 4.13: Diagramma di sequenza del funzionamento online della rete



Il primo strato avrà funzione di attivazione e propagazione identità, perciò entrambe le funzioni dovranno dare in uscita il valore d'ingresso (cioè il pattern). Perciò quando un pattern viene consegnato alla rete questo verrà elaborato dalle funzioni del primo strato. In figura 4.14 supponiamo X il vettore del pattern.

Figura 4.14: Funzioni dello strato d'ingresso

$$A_i^{in} = X_i \quad F(A_i)^{in} = A_i^{in}$$

Ogni neurone del primo strato ha ora dei valori di uscita da consegnare allo strato successivo.

Lo strato nascosto come già visto in analisi avrà funzione di attivazione con il compito di misurare la distanza fra il pattern d'ingresso e il vettore dei pesi del neurone. Questa distanza servirà per la funzione di propagazione a calcolare l'inverso della distanza più 1, così il neurone più vicino al pattern d'ingresso sarà quello con funzione di attivazione di questo strato minima, comportando una funzione di propagazione massima. Per un'analisi più approfondita sulla distanza euclidea si rimanda alla sezione 6.2.3.

In figura 4.15 supponiamo S vettore dei pesi e X valore del pattern.

Figura 4.15: Funzioni dello strato nascosto

$$A_i^n = d(X_i, S_i) \quad F(A_i)^n = \frac{1}{d(X_i, S_i) + 1}$$

Come già citato nella costruzione della mappa radio, ogni neurone dello strato d'uscita avrà pesi a 'zero' per i neuroni nascosti non collegati e pesi pari a 'uno' per quelli collegati. La funzione di attivazione sarà realizzata da una somma pesata in quanto un cluster nello strato nascosto può essere rappresentato da un solo neurone nello strato di uscita, ed un neurone dello strato di uscita può essere collegato ad un solo cluster dello strato nascosto.

Con questa tecnica essendoci alcuni pesi nulli la somma pesata verrà fatta solamente tra uscite di neuroni nascosti collegati. La funzione di propagazione sarà una soglia (da inserire come parametro della rete) con la

quale escludere neuroni di uscita che danno valori troppo bassi e perciò cluster lontani dal pattern. Nella figura 4.16 supponiamo ϑ la soglia.

Figura 4.16: Funzioni dello strato d'uscita

$$A_i^u = \sum_{i=1}^n S_i X_i \quad F(A_i)^u = A_i^u \text{ se } A_i^u > \vartheta \quad F(A_i)^u = 0 \text{ se } A_i^u < \vartheta$$

Da questo strato bisognerà scegliere il neurone che da uscita più alta, ovvero il cluster più quotato per essere il proprietario del pattern.

In seguito alle prove sul campo (vedi sez. 6.2.3) si noterà che la funzione di attivazione della rete non sarà proprio quella più adatta. Infatti la somma pesata rischia di incentivare la scelta verso cluster che possiedono più neuroni nascosti collegati. Infatti il peso consiste solo nel calcolare i soli collegati, mentre la somma incide sul numero dei collegati. Casi in cui il numero di neuroni collegati sono tanti, a causa di una maggiore quantità di rilevamenti, rischiano di compromettere la vittoria del neurone nascosto più vicino, magari unico nel suo cluster. Per evitare ciò verrà trasformata la funzione di attivazione, abolendo la somma, facendo passare il valore più alto in uscita dai neuroni nascosti così da avere una gara a parimerito fra tutti i neuroni nascosti della rete.

Anche per la fase di localizzazione, come per l'addestramento, ci si avvale dell'ordinamento automatico delle componenti di potenza dei pattern rilevate in un luogo. E' importante inserirle nello stesso ordine voluto dalla rete per non compromettere il suo mestiere.

4.2.2 Neurone

I pesi saranno una fondamentale caratteristica del neurone. Siccome in questo sistema i neuroni hanno solamente la caratteristica di avere pesi rispetto a quelli dello strato precedente si costruirà un'array per essi. Se si pensasse di utilizzare una rete fully connected dove i neuroni hanno pesi per altri di ogni strato sarebbe sufficiente costruire neuroni con una matrice al posto dell'array e una rete che ridefinisca il metodo di inizializzazione per avere neuroni più complessi.

Questi pesi dovranno poter essere modificati sia nel loro valore che nella loro quantità. I pesi saranno valori decimali, perciò non si deve permettere

all'utente di inserire valori scorretti. Ciò servirà nei casi in cui lo strato inferiore aumenti di uno il numero di neuroni.

Mentre per lo strato nascosto tutti i pesi avranno un valore decimale relativo ai neuroni d'ingresso a cui saranno tutti allacciati, nello strato d'uscita il valore dei pesi è sinonimo di collegamento con uno neurone dello strato nascosto. 1 è collegato, 0 altrimenti. Ovviamente non hanno senso i pesi per i neuroni d'ingresso.

Il parametro che definisce lo strato di appartenenza sarà un valore intero ove 0 è lo strato d'ingresso e così via a salire. Questo valore non si potrà modificare una volta creato il neurone.

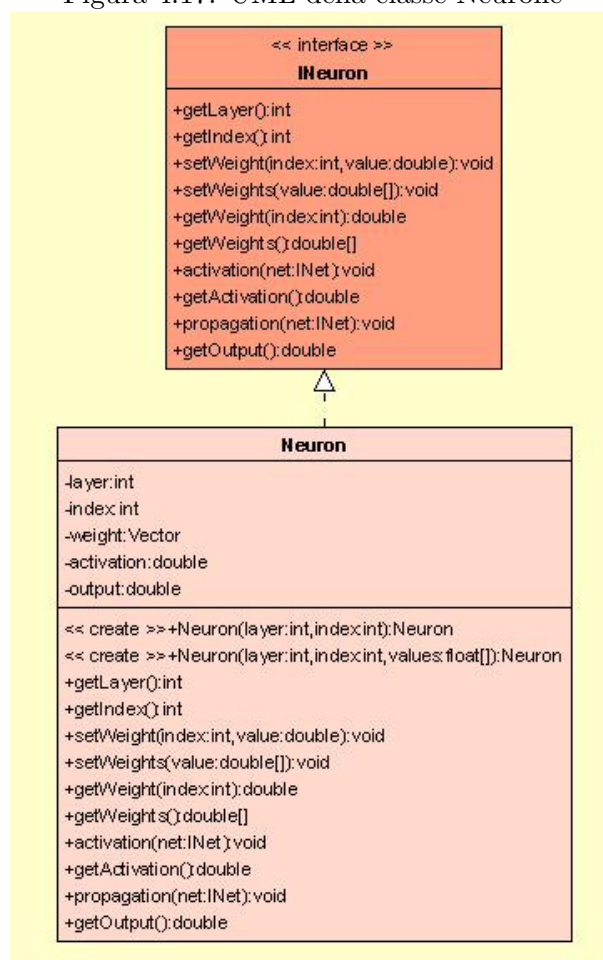
Oltre al parametro che definisce lo strato servirà un parametro che definisca in che punto dello strato ci si trova. Questo servirà nelle funzioni di attivazione per differenziare un neurone dall'altro. Si darà la possibilità di leggere questa informazione ma non di modificarla. Un neurone una volta posizionato in un indice non potrà spostarsi. Questo cabla in un certo senso la rete, ma in questo sistema, nel qual caso si desideri spostare un neurone, se ne dovrà costruire uno nuovo con la posizione voluta. Questo evita eventuali modifiche alla struttura non consentite.

Nella fase di localizzazione un neurone dovrà svolgere le sue funzioni perciò due ulteriori parametri saranno il valore delle funzioni di attivazione e propagazione (quest'ultima "uscita"). Infatti questi saranno calcolati in seguito all'esecuzione delle funzioni. Due metodi dovranno permettere di leggere questi dati dal neurone.

Il neurone metterà a disposizione due metodi per essere sollecitato a svolgere le funzioni sopra citate. Per ottenere un risultato eseguirà le funzioni di strato messe a disposizione dalla rete, comuni per tutti i neuroni dello stesso strato.

Nella figura 4.17 è riportato lo schema UML di questo componente.

Figura 4.17: UML della classe Neurone



4.2.3 Pattern

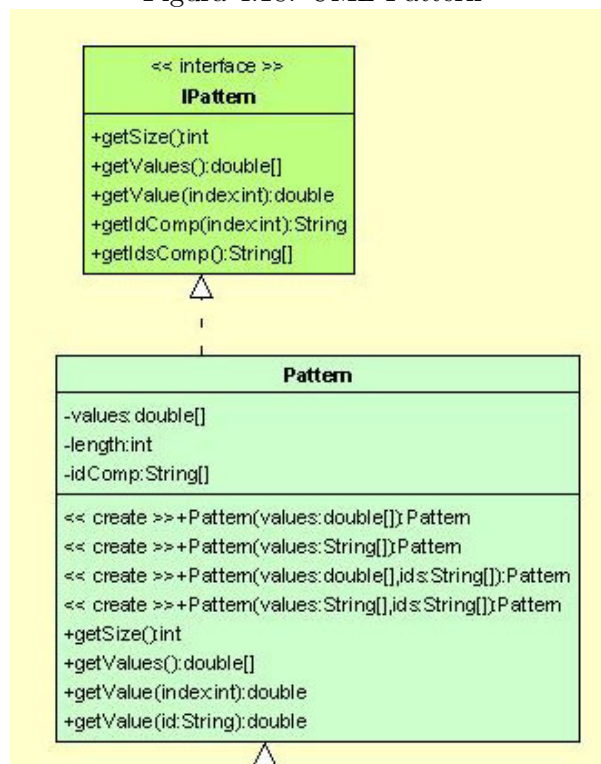
A differenza di quanto detto in analisi non si avranno due pattern fratelli uno per addestramento e uno per localizzazione perchè la tipologia di essi è la stessa. In realtà però con l'addestramento il pattern trasporta un'informazione in più, ovvero il cluster d'appartenenza. Come si vedrà in seguito il pattern utilizzato per l'addestramento sarà perciò una specializzazione di quello impiegato per la localizzazione.

Il patter avrà la caratteristica di possedere un array di valori (si ricorda che in questo sistema l'array risulta lungo quanto il numero di acces point).

Si fornisce la possibilità di reperire i valori del pattern dopo la creazione.

La figura 4.18 riporta la classe che rappresenta questo componente.

Figura 4.18: UML Pattern

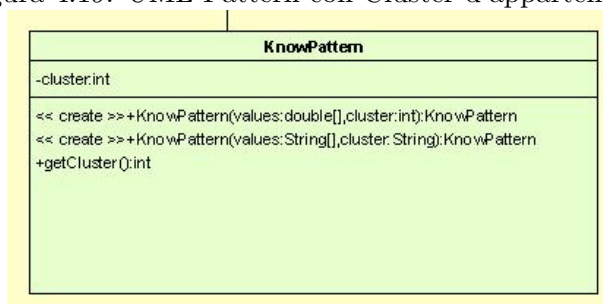


Per l'addestramento si pensa a pattern speciali. Questi saranno la spe-

cializzazione dei pattern standard ma con un campo in più. Infatti ogni pattern d'addestramento ha in sé il cluster al quale appartiene. Si differenziano i pattern di localizzazione da quelli d'addestramento ma non come due classi diverse. Quanto più come una la specializzazione dell'altra.

Di seguito riporta il diagramma UML.

Figura 4.19: UML Pattern con Cluster d'appartenenza



Infine i pattern sono stati potenziati con l'informazione inerente all'identificatore di ogni componente per poterli consegnare alla rete con lo stesso ordine degli access point a cui fanno riferimento. L'identificativo di cluster perde il valore simbolico ma assume un aspetto pratico grazie alla contraddistinzione eseguita tramite una stringa inserita dall'utente all'atto della prelevazione delle potenze.

4.2.4 Componenti del sistema

Il sistema si sviluppa quindi in tre principali componenti che dialogheranno tra loro attraverso un sistema lasco che dia la possibilità di intercambiare individualmente i pezzi dell'applicazione ottenendo una suddivisione in moduli di essa.

Un approccio ottimale potrebbe essere quello a "componenti" dove un sistema di dialogo a messaggi permetterebbe la distribuzione dei componenti nell'ambiente o l'intercambiabilità dinamica a run time. Per motivi temporali si farà una scelta più cablata analizzando in seguito quali veloci cambiamenti potrebbero portare ad una soluzione alternativa.

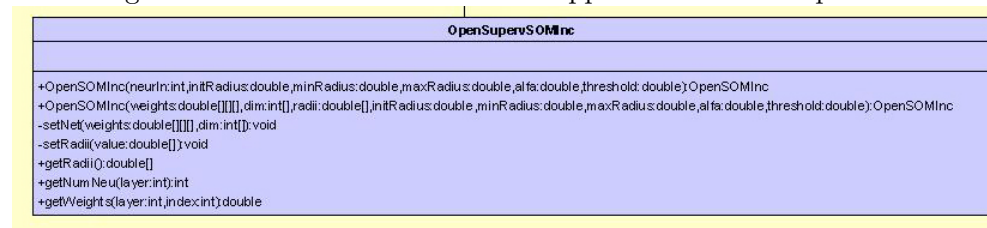
Rete "Open"

Questa rete sarà un'estensione di quella specializzata (vedi sez 4.2.1) che, come già citato, permetterà al sistema di accedere ai dati di essa in maniera aperta. Infatti i parametri della rete nel prototipo sviluppato sino ad ora sono rimasti protetti, mentre ora si intende permettere almeno la loro lettura, per dare la possibilità al sistema di visualizzare il suo stato.

Ci si accorgerà in realtà che questa rete dovrà permettere il salvataggio di struttura e dati e quindi anche un eventuale ripristino di essi.

La scelta consiste nel decidere se permettere la gestione di salvataggio alla rete o al gestore. Siccome la tipologia di salvataggio dati potrebbe cambiare modalità nel corso del tempo, non sarà certo compito la rete. Sarà parte del gestore a compiere ciò. Si riporta in fig 4.20 lo schema UML che meglio la approssima.

Figura 4.20: UML della classe che rappresenta la Rete aperta

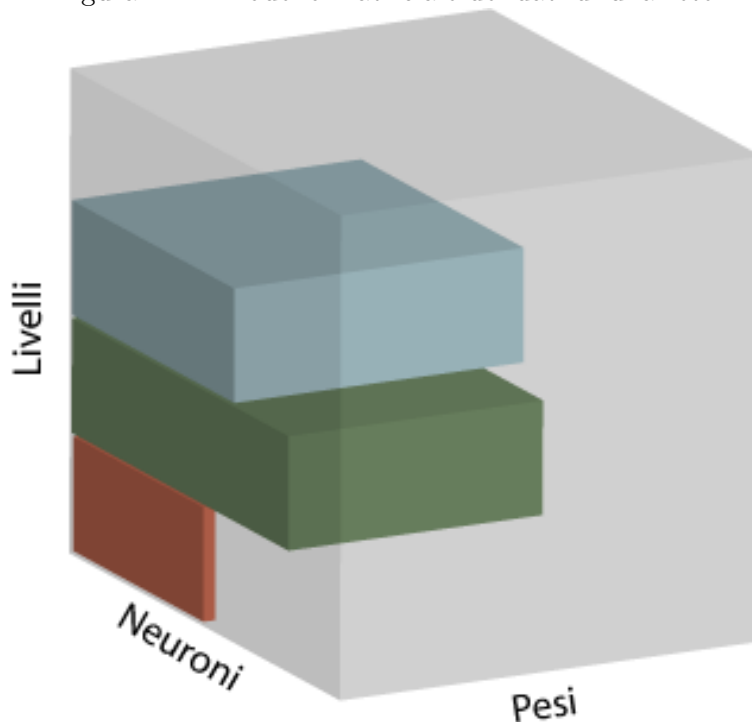


Come si può notare dal diagramma è possibile creare una rete in maniera standard (seguirà addestramento) o anche costruirla con tutti i pesi già definiti: si pensi ad una lettura di questi da file.

Grazie ai pesi disposti in una matrice tridimensionale si identificano già numero di livelli e numero di neuroni per ognuno. La matrice sarà minimo di altezza tre, mentre la larghezza dipenderà dallo strato con maggiore numero di neuroni e la profondità dal neurone con il maggior numero di pesi (comune per i neuroni di uno stesso strato).

Nella costruzione di una rete già addestrata, si opta di passare anche il numero di neuroni per ogni strato, così da prevenire onerosi calcoli per estrapolare le lunghezze degli array dalla matrice, visto che, nonostante una matrice abbia delle dimensioni fissate, strati diversi avranno numero di neuroni diversi.

Figura 4.21: Modello matriciale dei dati di una rete



Inoltre deve essere possibile settare i raggi dei neuroni nascosti. Per evitare che questi settaggi vengano eseguiti su una rete persistente non si permetterà all'utente di settare questi dati se non in costruzione.

Sarà inoltre possibile ottenere il numero di neuroni per ogni strato, e il peso di ogni neurone.

Console

La console dovrà disporre di pulsanti e campi dove inserire i valori per inizializzare la rete o effettuare addestramento e localizzazione su di essa. Questa prima console primitiva potrà essere in seguito sostituita con una più evoluta. La fig 4.22 ne riporta il probabile aspetto grafico.

La console sarà un componente da inserire in una finestra o in un'applet, e conterrà al suo interno le GUI per interagire con l'utente. Queste GUI verranno ascoltate da un apposito componente listener che potrà interagire con il gestore quando delle richieste verranno effettuate. Ricordiamo che a

Figura 4.22: Console grafica per l'amministratore della rete

The image shows a graphical user interface for a network administrator. It features a light blue background with several input fields and buttons. The input fields are arranged in a vertical list on the left, each with a corresponding label. The buttons are arranged in a grid-like fashion, with some buttons having a 3D effect. The labels for the input fields are: 'Numero Access Point', 'Parametro di modifica', 'Soglia d'uscita', 'Raggio iniziale', 'Raggio min', 'Raggio max', 'Pattern', and 'Cluster del Pattern'. The buttons are labeled: 'CREA', 'APRI', 'SALVA', 'ADDESTRA', 'LOCALIZZA', and 'ADDESTRA DA FILE'.

Numero Access Point	
Parametro di modifica	
Soglia d'uscita	
Raggio iniziale	
Raggio min	
Raggio max	
CREA	
APRI	SALVA
Pattern	
Cluster del Pattern	
ADDESTRA	LOCALIZZA
ADDESTRA DA FILE	

questo stato dell'applicazione della rete, essa potrà essere creata, addestrata a mano (pattern per pattern) o da file (formattato in xml con una struttura predefinita, vedi fig 5.6) e eseguire localizzazione. Infine si conta anche di poterla salvare e aprire da file.

Si punta a costruire una console che possa caricare un file formattato adeguatamente e inviare i suoi dati alla rete per l'addestramento. Questa soluzione mirerà a risolvere il problema di addestramento per un numero elevato di pattern. Nel capitolo successivo verrà sviluppata l'applicazione che permetterà di esportare un grande numero di dati riguardanti le potenze rilevate dall'ambiente per addestrare la rete.

Questa console mette insieme le funzionalità di addestramento e di localizzazione essendo uno strumento di utilizzo dell'amministratore del sistema. Una volta certificata la funzionabilità del sistema si potrebbe pensare di fornire console solo di localizzazione (e non di addestramento) a chiunque ne necessiti in rete.

Non si dimentichi che valori scorretti immessi nella console non verranno riconosciuti da essa al fine di non affidargli funzionalità non di sua competenza. Sarà la rete che avvertendo parametri mal posti comunicherà al gestore il tipo di errore, ed esso lo potrà comunicare all'utente attraverso un dispositivo di uscita.

La console verrà arricchita con un campo di testo che identifica la rete con un nome e una casella di dialogo che interagirà con l'amministratore all'atto della costruzione di una nuova rete. Questa sotto-interfaccia servirà per registrare gli access point che fanno riferimento ai neuroni d'ingresso della rete, in un determinato ordine.

Gestore rete

Il gestore della rete dovrà essere un ascoltatore dei comandi della console e dovrà eseguirli sulla rete, inoltre dovrà poter salvare i dati in un file e poter ricostruire una rete da questi.

Il gestore sarà quindi un componente vicino alla rete che si mette in ascolto dei comandi della console o per meglio dire del suo listener. Ovviamente in questa prima fase i comandi saranno chiamate a procedura ma si

potrebbe, solamente modificando il listener (visto come interfaccia al mondo della console), comunicare attraverso messaggi che un gestore più evoluto andrebbe a raccogliere in un ipotetico spazio.

In previsione di ciò si decide di passare al gestore solo comandi sotto forma testuale accompagnati da array di stringhe che rappresenteranno i parametri.

Per ora il gestore si limiterà a ricevere comandi da eseguire sulla rete attraverso un proprio metodo "do action" che definirà l'azione da compiere sulla rete quali la creazione di una nuova, dati i parametri da interfaccia, o l'addestramento e la localizzazione di pattern. Questo gestore oltre a ricevere i comandi e ad eseguirli, è l'unico che possa comunicare direttamente con la rete e con le viste per segnalare eventuali risultati, errori e rappresentazioni grafiche.

Non si dimentichi che il gestore potrà permettere un addestramento multiplo e veloce. Infatti si è pensato di inserire nell'applicazione, come già citato per la console, la possibilità di addestrare la rete attraverso l'ausilio di file dove saranno registrati più pattern di addestramento.

Questi pattern di potenza prelevati da vari punti dell'ambiente per tutti gli access point, con l'ausilio dell'applicazione sviluppata nel prossimo capitolo, saranno salvati su un file xml per poter essere letti dalla rete stessa. Sarà compito del modulo di gestione file xml interpretare il file e fornire i patterns per l'addestramento.

Il gestore sarà in grado anche di salvare lo stato della rete o crearne una attraverso file. Questo sarà un primo modo per memorizzare l'architettura della rete.

Per far ciò verrà utilizzato un modulo che gestisce file xml di facile interpretazione e accessibilità. Per permettere al gestore di rispondere alla localizzazione e magari qualsivoglia fornire una rappresentazione grafica della rete, si dovrà costruire una vista che esso potrà gestire, aggiungibile magari allo stesso pannello della console.

Modulo XML

Questo componente utilizzato dal gestore della rete verrà utilizzato nel caso l'utente voglia creare una rete con un suo stato già addestrato o almeno con

un'architettura già predefinita, per non dover ricorrere ad ogni utilizzazione del programma a reinizializzare e riaddestrare la rete.

Inoltre permetterà di salvare lo stato di una rete, ovviamente allo scopo di recuperarlo in un secondo momento.

In pratica quello che si memorizzerà sarà la stessa mappa radio che in altre applicazioni di scene analysis rappresenta una serie di potenze rilevate, mentre in questo caso è l'architettura della rete con all'interno i pesi che sono frutto di una elaborazione delle potenze inserite come patterns d'addestramento.

Nel pattern MVC questo rappresenta il model dell'applicazione ovvero quella parte del sistema (la rete) che ha in se la mappa radio, la quale viene salvata per essere riutilizzata, e modificata.

Senza questa parte non si potrebbe avere un sistema completo senza ogni volta dover ricostruire da zero la mappa radio e rieseguire l'intero addestramento.

La prima azione di questo modulo, ovvero quella di salvataggio consisterà nell'utilizzare un parser, offerto già da java per la gestione di file xml. Nel file dovranno andare registrati:

- I parametri costanti della rete
 1. Raggio iniziale
 2. Raggio max
 3. Raggio min
 4. Learning Rate
 5. Soglia
- I parametri dell'architettura di rete
 1. Numero di neuroni dello strato d'ingresso
 2. Numero di neuroni dello strato nascosto
 3. Numero di neuroni dello strato d'uscita

- I valori numerici
 1. L'indirizzo MAC degli Access Point rappresentati dai neuroni d'ingresso
 2. I valori dei raggi dei neuroni nascosti
 3. I valori dei pesi dei neuroni nascosti
 4. I valori dei pesi dei neuroni d'uscita
 5. Il numero dei pattern d'addestramento inseriti per ogni neurone di uscita (cluster)

Tutti questi parametri che il gestore può ottenere dalla rete, saranno passati al modulo xml per essere scritti su file. In figura 4.23 si riporta un esempio di rete addestrata con 3 neuroni d'ingresso, 1 nascosto e 2 d'uscita, che chiarifica la sintassi utilizzata.

Un altro compito del modulo per files XML, a cui il gestore della rete fa riferimento per memorizzare dati in memoria, sarà quello di prelevare dei pattern da un file di rilevazioni di potenza e fornirlo al gestore per eseguire addestramento di rete.

Questi file .xml conterranno un identificatore di cluster e uno di pattern (o meglio del punto di prelevazione nel cluster) oltre a tutti i valori di potenza espressi in dBm, catturati dai dispositivi wireless presenti in rete (vedi fig 5.6).

Per addestrare la rete a questo stato dell'arte non serviranno gli identificativi di pattern ma serviranno soprattutto quelli di cluster per saper a quale il pattern appartiene. I primi potrebbero essere utilizzati da altre applicazioni nel qual caso si voglia ottenere una localizzazione più fine a livello di punto e non solo di stanza.

Da non sottovalutare la questione degli access point. Infatti la lettura da file fisserà un certo ordine nelle potenze ricevute uguale per ogni rilevazione ma che potrebbe non coincidere nell'ordine con un altro file o un pattern inserito manualmente. Si confida ancora nella accuratezza dell'amministrazione nel controllare che i pattern che addestrano la rete abbiano sempre lo stesso ordine degli access point ai quali le potenze fanno riferimento.

Figura 4.23: Esempio file XML per salvataggio rete

```
<?xml version="1.0" encoding="UTF-8"?>
<net name="Rete SOM supervisionata">
  <init-parameters>
    <init-radius>1.0</init-radius>
    <min-radius>0.0</min-radius>
    <max-radius>2.0</max-radius>
    <alfa>0.3</alfa>
    <threshold>0.3</threshold>
  </init-parameters>
  <input-neurons>
    <neu-0 />
    <neu-1 />
    <neu-2 />
  </input-neurons>
  <hidden-neurons>
    <neu-0>
      <radius>1.0</radius>
      <weights>
        <wei-0>1.0</wei-0>
        <wei-1>2.0</wei-1>
        <wei-2>3.0</wei-2>
      </weights>
    </neu-0>
  </hidden-neurons>
  <output-neurons>
    <neu-0>
      <weights>
        <wei-0>0.0</wei-0>
      </weights>
      <numTrainings>0</numTrainings>
    </neu-0>
    <neu-1>
      <weights>
        <wei-0>1.0</wei-0>
      </weights>
      <numTrainings>1</numTrainings>
    </neu-1>
  </output-neurons>
</net>
```

Per minimizzare il rischio di inserire pattern mal ordinati si opterà di togliere all'amministrazione la possibilità di addestrare manualmente la rete, lasciando come unica alternativa l'addestramento da file.

Si decide invece di mantenere la possibilità di localizzare da console un pattern di potenze. Questo per testare velocemente il funzionamento della rete. Si confida per questo nell'attenta inserzione ordinata dei valori di potenza da parte dell'utente, ricordando che i neuroni d'ingresso sono associati univocamente ognuno ad un access point.

Viste

Le viste possono essere tante e di diversa tipologia. Queste viste potranno essere contattate dal gestore grazie ad un metodo di aggiornamento messo a disposizione da esse. Si decide di non abbandonare la gestione a messaggi (comandi), per poter tener le porte aperte ad una locazione distribuita anche delle entità gestore e vista.

La più semplice vista, sviluppata inizialmente, sarà quella inerente alla risposta di richiesta di localizzazione, ovvero il risultato che rete fornisce (cluster vincitore). In realtà questa vista non sarà altro che un visualizzatore di stringhe, quelle che il gestore decide di trasmettergli.

Una seconda vista che si intende sviluppare sarà quella che rappresenta lo stato della rete in veste grafica. Si potranno graficare i neuroni e i loro relativi pesi e raggi. Una veste più specializzata potrà visualizzare altri dati inerenti ad ogni neurone, quali l'access point di pertinenza per quelli d'ingresso e il cluster d'appartenenza per quelli di uscita.

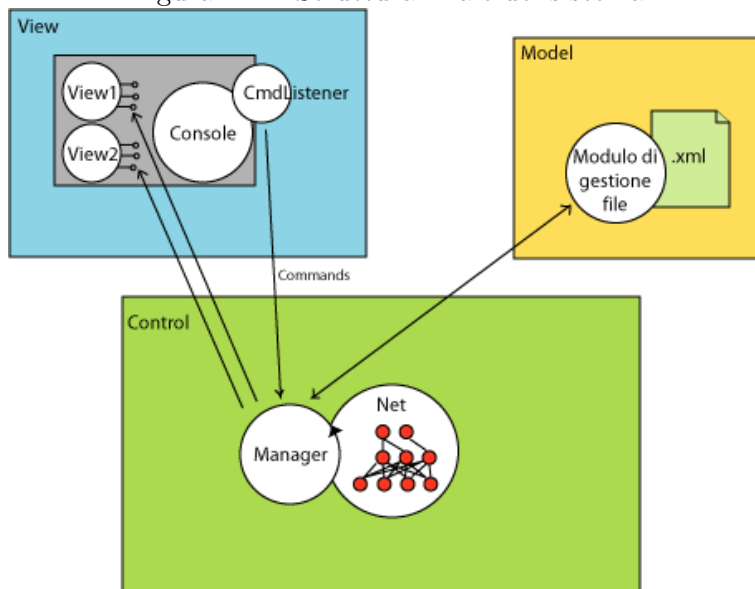
Struttura finale

Lo schema di figura 4.24 permette di raffigurare ogni parte del sistema in costruzione.

4.3 Architettura finale

Il miglior modo per rappresentare il prototipo dell'applicazione risulta quello di mostrare il diagramma UML delle classi sviluppate in java. Come

Figura 4.24: Struttura finale del sistema



si può notare dalla fig 4.25 sono state suddivise le classi in base alla loro appartenenza ai moduli MVC.

4.4 How to use

In questo paragrafo si intende avvicinare l'utente all'applicazione e perciò al suo utilizzo. Come prima anticipazione viene mostrata l'interfaccia grafica dell'applicazione con la console e le viste in uno stato di già avvenuto addestramento.

Si può notare dalla figura 4.26 che è stata aperta una nuova rete già addestrata. Per comprendere meglio il significato dei campi si è deciso di trascrivere in essi i valori inerenti alla rete.

Per poterla addestrare ulteriormente basterebbe farlo cliccando sul pulsante d'addestramento e selezionando un file .xml con all'interno nuove rilevazioni di potenza.

Si può notare che l'applicazione dispone di un menu per svolgere azioni sulla rete, ottenere aiuti per l'utilizzo della rete o addirittura visualizzare statistiche di essa. E' possibile inoltre inserire pattern per effettuare la loro localizzazione, mentre sotto la console risiedono due finestre di output, una per comunicare le mansioni svolte dalla rete o problemi scaturiti da mal

Figura 4.25: Architettura finale del sistema

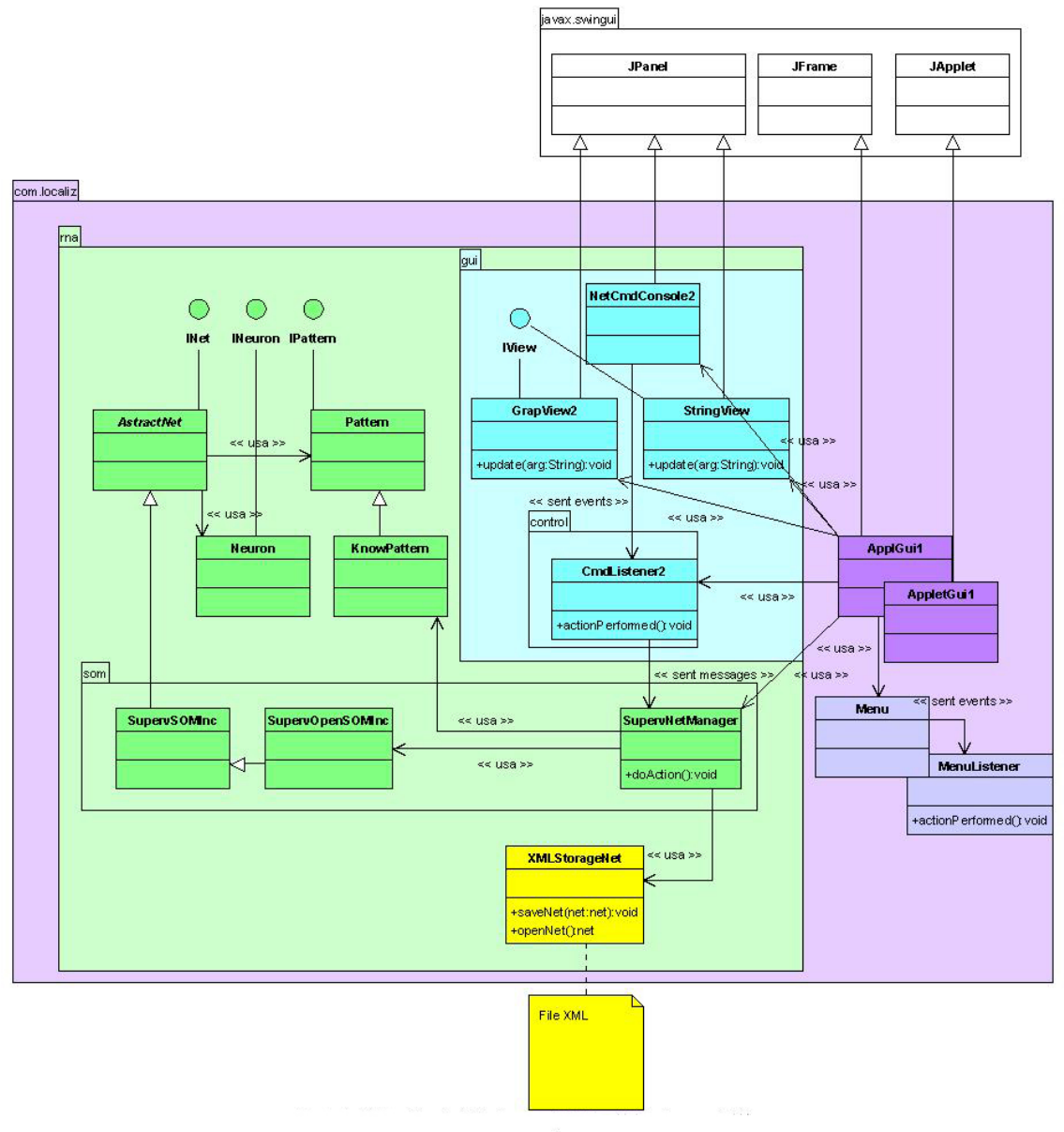


Figura 4.26: Gui dell'applicazione rete neuronale

Localizzazione

Option Help Statistics (it)

Nome della rete	venezia
Numero di Access Point	4
Raggio iniziale del neurone	1
Dimensione minima Raggio	0.4
Dimensione massima Raggio	2.5
Learning rate (<0.5)	0.03
Soglia di esclusione clusters (piccolo! <0.5)	0
Crea rete (richiesta MAC)	
Apri Rete	Salva Rete
Inserire pattern da localizzare (es: 2,2.3,1,5.23)	
Addestra da file	Localizza pattern

Messaggi rete:

New Net Loaded from C:\Documents and Settings\aggshow\Desktop\Dati Temporanei Tesi\prove\primo piano rete.xml

Rete Neuronale

Name: venezia Init Radius: 1.0 Min Radius: 0.4 Max Radius: 2.5 Learning Rate: 0.03 Threshold: 0.0

 Name: studio1	 Name: studio2	 Name: studio3
1.0 1.0 0.0 0.0 0.0	0.0 0.0 1.0 0.0 0.0	0.0 0.0 0.0 1.0 1.0

 r: 1.0	 r: 1.0	 r: 1.0	 r: 1.0	 r: 1.0
-63.0 -89.0 -87.8 -87.2	-67.9 -86.3 -83.4 -82.3	-66.1 -88.6 -80.0 -85.0	-54.9 -85.9 -85.1 -82.8	-63.3 -87.1 -83.0 -84.5

			
0:64:5:82:226:4	0:64:5:82:226:0	0:64:5:91:89:140	0:64:5:82:226:12

funzionamento di essa, e l'altra per visualizzare graficamente il suo stato attuale.

Capitolo 5

Prelevamento di pattern di potenza

5.1 Caratteristiche dell'ambiente

L'ambiente in cui ci troviamo sarà senza dubbio quello di un locale chiuso, che però dovrà comprendere anche quelle zone esterne limitrofe all'edificio. Quest'ultimo, dovrà senza meno essere rifornito di access point per eseguire la localizzazione, gli stessi access point che permettono agli utenti di connettersi ad internet. L'edificio potrebbe essere composto da grandi stanze contenenti anche più access point, oppure piccoli studi che sfruttano gli AP presenti nei corridoi. La casistica sarà vasta, anche se per questo progetto si tende pensare un edificio con piccole e numerose stanze così da diversificare maggiormente i valori di potenza rilevati.

Infatti si dovranno rilevare le potenze in più punti dell'edificio per darli in input come pattern alla rete neurale. In seguito alle caratteristiche dell'edificio avverrà un settaggio differente della rete, ma questo sarà oggetto del prossimo capitolo.

5.2 La tecnica di prelevamento

Il prelevamento delle potenze dai numerosi access point rilevabili in un punto dell'edificio, avverrà attraverso driver del sistema operativo (per questa tesi si considera il sistema operativo Windows XP SP1) comunicanti diretta-

mente con l'applicazione. In questo capitolo si intende spiegare come verrà sviluppata l'applicazione definendo input quel che si riceverà dall'ambiente e output quel che si dovrà fornire per addestrare la rete neuronale e non solo. Si pensi infatti come questa applicazione possa essere riutilizzata anche per fornire segnali di potenza ad altri sistemi di localizzazione. In questa maniera si potranno anche confrontare tecniche diverse su rilevazioni identiche.

Come già citato nei primi capitoli, gli access point emettono delle potenze che quando vengono ricevute dalle antenne wireless mobili hanno valori in funzione del punto in cui si trovano. Questo è il principio su cui si basa la tecnica di Scene Analysis, essa funzionerà tanto meglio quanto più sono le potenze rilevabili dai mobili (ovvero maggiore è il numero di access point che le emettono). Perciò, definito lo scopo di prelevare le potenze di tutti gli AP rilevabili, bisogna comprendere che non basterà un'unica rilevazione in un sol punto. Infatti l'andamento delle potenze rilevate non sarà certo costante ma è variabile (anche il solo spostamento di un uomo nelle vicinanze può variare la potenza), in seguito a numerosi fattori ambientali e artificiali. Per superare questo problema si intraprende la scelta di compiere numerose letture di potenza, e in seguito di farne una media. Le prelevazioni verranno fatte in punti ben precisi, ma in tutte quattro le direzioni cardinali. Questo infatti per ovviare il problema del rumore causato dalla presenza dell'amministratore che effettua la rilevazione. Quindi si avranno quattro medie per punto, ognuna rappresentante una direzione (N-W-S-E), mediate in seguito fra loro.

5.3 L'applicazione

Di seguito, seguendo una corretta metodologia ingegneristica, si definiranno passo per passo tutte le fasi al fine di ottenere un'applicazione robusta e funzionante.

5.3.1 Analisi

Si parte dalla presenza di access point nell'edificio; l'applicazione dovrà in qualche maniera prelevare dall'antenna della scheda wireless presente sul mobile una serie di potenze da elaborare come già definito.

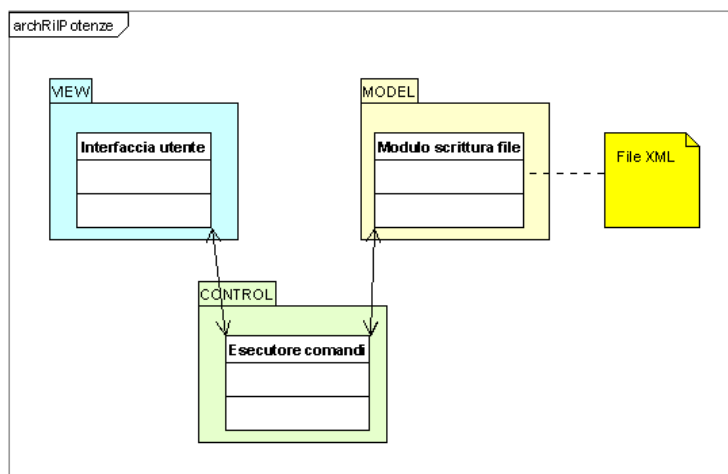
Volendosi soffermare sulla prelevazione delle potenze e non sulla loro elaborazione, è chiaro che la difficoltà consiste nell'operare a basso livello per prelevare dall'antenna i dati che interessano. Si dovrà fare uso di driver adeguati, che possano cooperare con librerie del sistema operativo. Queste ultime forniscono delle funzioni per le applicazioni a livello superiore. E' chiaro quindi che anche se l'applicazione potesse girare su un altro sistema operativo necessiterebbe di un'interfaccia diversa con il componente hardware che riceve la potenza.

Una volta posseduti i mezzi per dialogare con l'hardware l'applicazione sarà composta da un'interfaccia per l'utente che lo guidi a prelevare i dati dall'ambiente, e un modulo per la scrittura di questi dati su file, in un formato che possa essere letto non solo dal sistema che questa tesi descrive ma magari anche da altri sistemi concorrenti.

5.3.2 Progetto

Abbiamo quindi definito due moduli per questo sistema di raccolta dati. Il primo quello riguardante l'interazione con l'utente che compie le rilevazioni nell'ambiente, il secondo inerente la scrittura dei dati sensibili su file.

Figura 5.1: Architettura dell'applicazione



Per fare ciò ci si avvale ancora del linguaggio Java, multiplatforma e

flessibile, anche se in questa situazione, come già descritto in precedenza, la possibilità di operare su più piattaforme è nulla a causa dell'interazione con l'hardware specifica da sistema a sistema. Dovendo scegliere un sistema, si punta su quello sicuramente più utilizzato (Windows XP), mentre per l'interazione con i driver si riutilizzerà il lavoro WRAPI [Microsoft Research, <http://sysnet.ucsd.edu/pawn/wrapi/>] e JWRAPI [Johan Kristiansson, <http://media.csee.ltu.se/~johank/javawrapi/>].

Le prime sono una serie di DLL che in seguito all'installazione del protocollo Raw Packet NDIS e alla sospensione del servizio Wireless Zero Configuration di Windows forniscono funzioni per prelevare diverse informazioni dalla rete wireless e dagli AccessPoint rilevabili. Di queste informazioni quelle che utili al progetto saranno: - MAC ADDRESSES degli access point - SEGNALE DI POTENZA degli stessi - NOME della rete Queste DLL permettono inoltre altre funzionalità che però verranno sorvolate in questa fase.

Le JWRAPI sono invece una "javizzazione" delle WRAPI, grazie al package Java Native Interface fornito dalle distribuzioni Java è possibile costruire una DLL che faccia da ponte fra linguaggio C (linguaggio in cui sono state scritte le DDL WRAPI) e linguaggio Java, ovvero richiamare funzioni di essa come se fossero del linguaggio stesso.

Come già visto da figura 5.1 questa applicazione seguirà il pattern MVC, ove ogni modulo sarà rappresentato da una classe di Java; in seguito si descrivono i tre moduli rappresentati ciascuno da una classe.

Il primo modulo risulta essere quello direttamente a contatto con l'utente, l'interfaccia grafica. Realizzata con le GUI di Java essa dovrà avere un aspetto simile alla figura 5.3

Come si può notare, l'utente dovrà inserire il cluster di appartenenza del punto rilevato, si pensi ai pattern d'addestramento della rete neurale supervisionata: essi infatti portano con se l'informazione del cluster d'appartenenza. Si può pensare che ogni punto prelevato con una media delle potenze sui quattro lati per ogni access point sia un pattern da dare in addestramento alla rete. Per avere un rilevamento più completo e compatibile con le diverse tecniche di localizzazione, oltre a specificare il cluster d'appartenenza, bisogna fornire un identificativo del punto (ad esempio grazie ad una

Figura 5.2: Dall'applicazione allo strato fisico

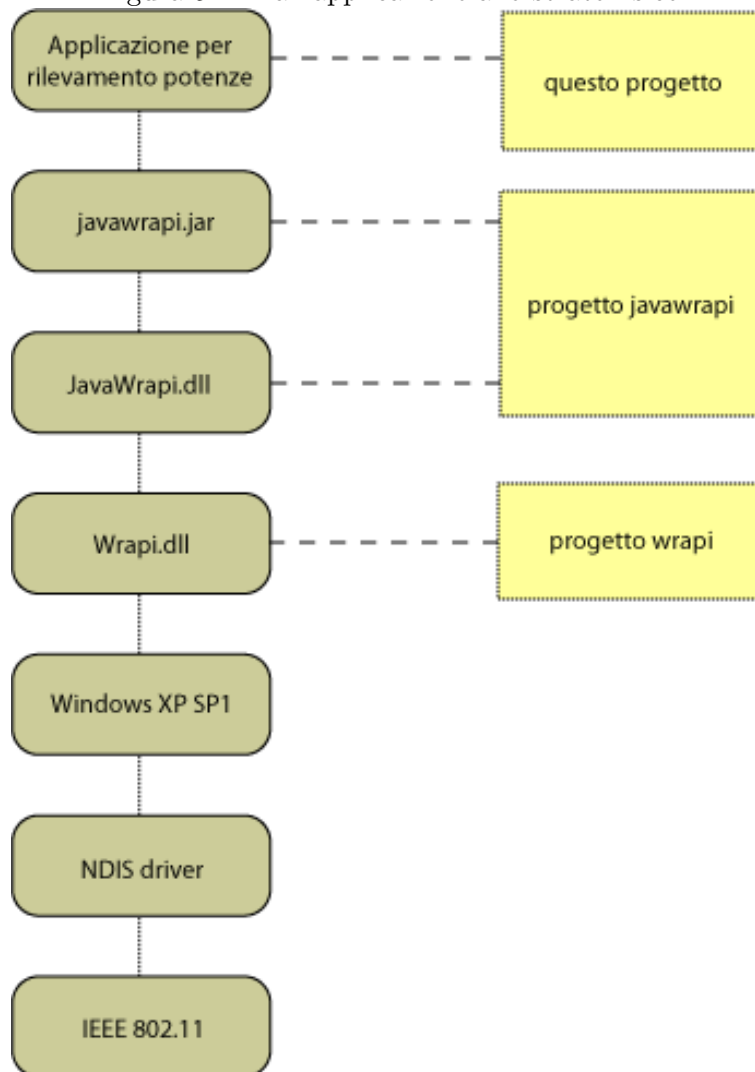


Figura 5.3: Bozza d'interfaccia prelevamento potenze



La bozza d'interfaccia, intitolata "RILEVAMENTO POTENZE", presenta tre campi di input: "Nome Cluster:" con il valore "Studio 1", "Id Rilevamento nel Cluster:" con il valore "03", e "Numero misure per lato:" con il valore "1000". Sotto questi campi, una sezione "Output:" contiene un riquadro con le istruzioni: "Ruotati verso N...", "Prelevate le misure da N.", e "Ruotati verso E...". In basso, un pulsante blu con il testo "Preleva 1000 misure per lato E" è pronto per essere premuto.

sequenza numerica). Per la tecnica basata sulla rete neurale questo parametro potrebbe venire utilizzato per effettuare una localizzazione più fine, o comunque per aiutare la rete a fornire un risultato corretto potendo specificare anche come l'utente si muove nel cluster e perciò conoscere con più facilità il cluster successivo, o comunque il più probabile ad esserlo. Per altre applicazioni servirà a distinguere i punti di prelevamento potenze all'interno di uno stesso cluster. Si potrebbe pensare anche ad una media di questi se la tecnica necessitasse di un punto per cluster.

Il numero di misure da compiere per ogni punto del cluster sarà il numero di misure per lato moltiplicate per 4 considerando che si effettueranno tutte le misure per ogni lato. Infatti come già accennato per minimizzare problemi di fading dovuti alla persona che tiene in mano il mobile, e per avere una più accurata rilevazione su entrambi i punti cardinali, si effettueranno un numero di misure per quattro lati e in seguito una media di tutte per ottenere il valore di potenza fornita da ogni access point.

L'interfaccia grazie ad una piccola finestra testuale di output comunicherà all'utente quando cambiare lato per effettuare una rilevazione per lo stesso punto ma in direzione diversa, e quando l'intero punto è stato registrato. Un grosso tasto invece informerà l'applicazione di eseguire le rilevazioni

per un lato. Nella figura d'esempio l'utente ha già effettuato 1000 rilevazioni da nord e si sta apprestando a compierle a est.

Una classe non citata ma ovviamente esistente sarà quella in ascolto dei comandi della console per gestirla e salvare i dati su file, attraverso un ulteriore componente per la scrittura. Il comandi che l'utente potrà impartire è la scansione del punto alla ricerca delle potenze degli access point, per tutti e quattro i lati. Questo dopo aver definito i parametri di ricerca, già citati.

Perciò l'ascoltatore dell'interfaccia dovrà interagire comunicando con l'utente quando doversi ruotare e quando tutte le rilevazioni per un punto saranno terminate. Quando si sarà giunti al termine, l'esecutore di comandi si ritroverà con un numero di rilevazioni totali per lato per access point del punto. Infine i dati sono tutti da mediare in un unico valore per ogni access point e da scrivere su file.

Si intende fare due fasi di mediazione per velocizzare il calcolo. La prima fase consisterà nel mediare tutti i valori di un lato, e la seconda fase di mediare i quattro valori (array di valori considerando potenze da più access point) per lato.

Prima di fare mediazioni però si devono fare considerazioni sulle potenze. Siccome queste vengono fornite in dBm, per fare una media di queste occorre tornare ad avere valori lineari (milliWatt). Inoltre essendo dei valori numerici molto piccoli, conviene a maggior ragione mediare dei microWatt. Perciò la mediazione avverrà attraverso i seguenti passi:

1. Ogni valore in dBm verrà trasformato in dB μ ($\text{dB}\mu = \text{dBm} + 30$)
2. Ogni dB μ verrà trasformato in μW ($\mu\text{W} = 10^{\frac{\text{dB}\mu}{10}}$)
3. Con i μW si farà la media per tutte e quattro le direzioni
4. In seguito per ogni access point si tradurranno di nuovo μW in dB μ ($\text{dB}\mu = 10\log_{10}\mu\text{W}$)
5. Infine si torna con valori espressi in dBm ($\text{dBm} = \text{dB}\mu - 30$)

L'applicazione riempirà, ad ogni punto monitorato, una lista degli effettivi access point rilevati, incrementando questa lista ad ogni scoperta di nuovi access point. Questa lista servirà per scrivere su file i valori per punto per ogni access point. Con la possibilità di aggiornare un file preesistente, nel caso le rilevazioni non possano essere effettuate tutte in una singola esecuzione, questa lista dovrà essere aggiornata dei mac già presenti su file, per poter riscrivere il file con tutto il database completo. Perciò da quanto detto si deduce che l'aggiornamento di un file preesistente consisterà nella lettura e nella riscrittura di esso, questo per ovviare a problemi di accesspoint non rilevati. Infatti si dovrà dare valori di potenza $-\infty$ dBm, in realtà un valore molto basso, per quegli access point che in certi punti non sono stati rilevati.

Verrà costruita perciò una struttura, che permetta di raccogliere tutti i dati delle rilevazioni, persino da file; questa struttura non sarà ridondante perchè conterrà solo i valori di potenza effettivamente rilevati. Sarà poi nella trascrizione su file che grazie alla lista di access point verranno inseriti i valori di potenza che rappresentano $-\infty$. La struttura controllata dall'applicazione a runtime si presenterà come in figura 5.4, dove abbiamo 4 record, con ognuno il numero di propri access point rilevati e i dati inerenti al punto di prelievo.

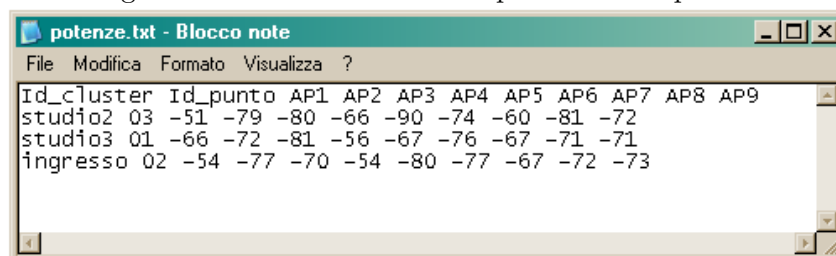
Figura 5.4: Struttura rilevazioni

IdCluster	IdPoint	MAC	Power	MAC	Power	MAC	Power	MAC	Power	MAC	Power
IdCluster	IdPoint	MAC	Power	MAC	Power	MAC	Power	MAC	Power		
IdCluster	IdPoint	MAC	Power	MAC	Power	MAC	Power	MAC	Power	MAC	Power
IdCluster	IdPoint	MAC	Power	MAC	Power	MAC	Power				

Il modulo mancante sarà quello riguardante la gestione di file. Questo infatti dovrà salvare su file una tabella di valori rappresentanti le potenze rilevate dagli access point per ogni punto. Di seguito si riporta un ipotetica tabella che definisce il probabile database su file di testo.

Su file si vorrebbe permettere di scrivere dati in due modalità, testo e xml. Nel progetto si svilupperà per prima la soluzione di salvataggio su file di testo, per fornire al mondo una visione standard della mappa radio,

Figura 5.5: Bozza d'interfaccia prelevamento potenze



The screenshot shows a Notepad window with the title 'potenze.txt - Blocco note'. The menu bar includes 'File', 'Modifica', 'Formato', 'Visualizza', and '?'. The text area contains a table with power measurements for three clusters: studio2, studio3, and ingresso. Each cluster has 9 access points (AP1 to AP9) and a corresponding 'Id_punto' value.

Id_cluster	Id_punto	AP1	AP2	AP3	AP4	AP5	AP6	AP7	AP8	AP9
studio2	03	-51	-79	-80	-66	-90	-74	-60	-81	-72
studio3	01	-66	-72	-81	-56	-67	-76	-67	-71	-71
ingresso	02	-54	-77	-70	-54	-80	-77	-67	-72	-73

libera da tag che potrebbero complicarne la lettura se letti con strumenti non dotati di parser appositi.

Si concede inoltre all'utente la possibilità di salvare misurazioni su nuovo file e di caricare misurazioni da file esistente. In realtà caricare informazioni nell'applicazioni a prima vista non sembra interessante, si sarebbe potuto ovviare appendendo al file esistente le ultime misurazioni effettuate. Ma c'è una considerazione da fare. Si corre il rischio di appendere ad un file misure che non hanno rilevato tutti gli access point rilevati dalle misure precedenti del file. Perciò occorre caricare tutte le misurazioni da file e riscriverle su file (riscegliendo il salvataggio dei dati) cosicchè verranno definite anche le misurazioni per access point non rilevati (avranno un valore prossimo a $-\infty$ dB). Si dà così la possibilità all'utente anche di scegliere se salvare su nuovo file o aggiornare quello caricato riscegliendolo per il salvataggio. Un messaggio di testo comunicherà quante misurazioni l'applicazioni sta gestendo e si appresterà a salvare.

In seguito si è preferito sviluppare anche una tipologia di salvataggio e caricamento per file xml. Infatti questa struttura più efficiente e riconosciuta oramai in ogni ambito scientifico permette facilmente all'applicazione della rete neurale già sviluppata di recuperare i pattern per essere addestrata. Essa infatti utilizza il linguaggio xml per effettuare addestramento da file e senza inserzione manuale. Grazie a queste due tipologie sarà possibile confrontare diverse tecniche di localizzazione con gli stessi valori di potenza; infatti tecniche come RADAR prediligono la lettura di potenze da file di testo. In figura 5.6 si mostra quale sarà il risultato di una serie di rilevazioni salvate su file XML.

Figura 5.6: Esempio di come vengono salvate le potenze su un file XML

```
<?xml version="1.0" encoding="UTF-8"?>
<file title="Rilevamento potenze access points">
  <MAC-addresses>
    <AP-0>0:13:136:152:204:46</AP-0>
    <AP-1>0:64:5:82:226:0</AP-1>
    <AP-2>0:64:5:91:89:140</AP-2>
    <AP-3>0:64:5:82:226:4</AP-3>
    <AP-4>0:64:5:82:226:12</AP-4>
  </MAC-addresses>
  <Strengths>
    <Rec-0 cluster="studiol" point="scrivania">
      <AP-0>-87.0</AP-0>
      <AP-1>-90.1</AP-1>
      <AP-2>-82.3</AP-2>
      <AP-3>-63.0</AP-3>
      <AP-4>-98.1</AP-4>
    </Rec-0>
    <Rec-1 cluster="studiol" point="finestra">
      <AP-0>-84.1</AP-0>
      <AP-1>-80.0</AP-1>
      <AP-2>-88.0</AP-2>
      <AP-3>-73.5</AP-3>
      <AP-4>-94.9</AP-4>
    </Rec-1>
    <Rec-2 cluster="studio3" point="finestra">
      <AP-0>-81.0</AP-0>
      <AP-1>-93.2</AP-1>
      <AP-2>-78.5</AP-2>
      <AP-3>-93.1</AP-3>
      <AP-4>-70.2</AP-4>
    </Rec-2>
    <Rec-3 cluster="biblioteca" point="tavolo">
      <AP-0>-97.2</AP-0>
      <AP-1>-86.0</AP-1>
      <AP-2>-84.0</AP-2>
      <AP-3>-63.8</AP-3>
      <AP-4>-90.9</AP-4>
    </Rec-3>
  </Strengths>
</file>
```

5.3.3 Prodotto - How to use

Il risultato finale sarà pertanto un'applicazione Java che un utente incaricato del servizio di localizzazione periodicamente utilizzerà per aggiornare la mappa radio.

Questa applicazione fornisce all'amministratore la possibilità di inserire cluster e identificativo punto nel cluster e, ruotando nelle quattro direzioni cardinali, prelevare campioni di potenza mediati per fornire un valore per ogni access point. Questa rilevazione del punto potrà così essere salvata su file, ad esempio in seguito ad una raccolta di altre misurazione da altro file, per poter fornire alle tecniche di cluster analysis, come quella discussa in questa tesi, valori reali da elaborare.

Purtroppo c'è da citare che l'applicazione non è solamente dipendente dal sistema operativo ma anche dall'hardware. Infatti le potenze prelevate da laptop a laptop (unici dispositivi al momento utilizzati per eseguire test) possono variare. Le potenze sono soggette a variazioni dovute alle caratteristiche tecniche delle schede wireless e dell'hardware sul quale sono montate.

Dalla fonte [10] è emerso uno studio fatto per comprendere questa diversità. Si è scoperto che l'unica caratteristica che può accomunare due antenne differenti è la forma della funzione densità di probabilità delle potenze. Quindi una soluzione che sarebbe possibile adottare risulterebbe quella di settare l'applicazione prima del prelievo delle potenze, al fine di normalizzare i valori rilevati a seconda di quale hardware si utilizza.

In questo progetto la macchina utilizzata sarà soltanto una, sia nella fase offline che nella fase online, così da non incorrere su problematiche dovute alla diversità dell'hardware.

Capitolo 6

Testing del sistema di localizzazione

Si decide di testare il sistema prima di sviluppare l'applicazione come servizio agli utenti. Infatti l'applicazione sviluppata sino ad ora rappresenta quella parte che verrà gestita dall'amministrazione ma contiene già in se i componenti necessari a fare localizzazione di pattern. Si pensi perciò che all'attuale stato dell'arte è possibile rintracciare la posizione di un mobile date le potenze che questo riceve dagli access point. Pensando tutta l'applicazione su un solo mobile, una volta fatto addestramento della rete è possibile localizzarlo. Ovviamente, non essendo ancora in un ambito distribuito, sarà lo stesso mobile che contiene la rete a dover essere localizzato da essa. In seguito con la trasformazione dell'applicazione in servizio si potranno ottenere funzionalità quali la localizzazione di un utente remoto o dello stesso che ne fa richiesta.

Questa esperienza aiuta a testare la rete neurale e definire quali sono i valori più adatti per i propri parametri dipendenti dall'edificio su cui si opera ma in qualche misura determinabili con considerazioni generalizzate (vedi sez 6.2). In realtà oltre ai parametri potranno essere ritoccati perfino alcuni algoritmi della rete stessa.

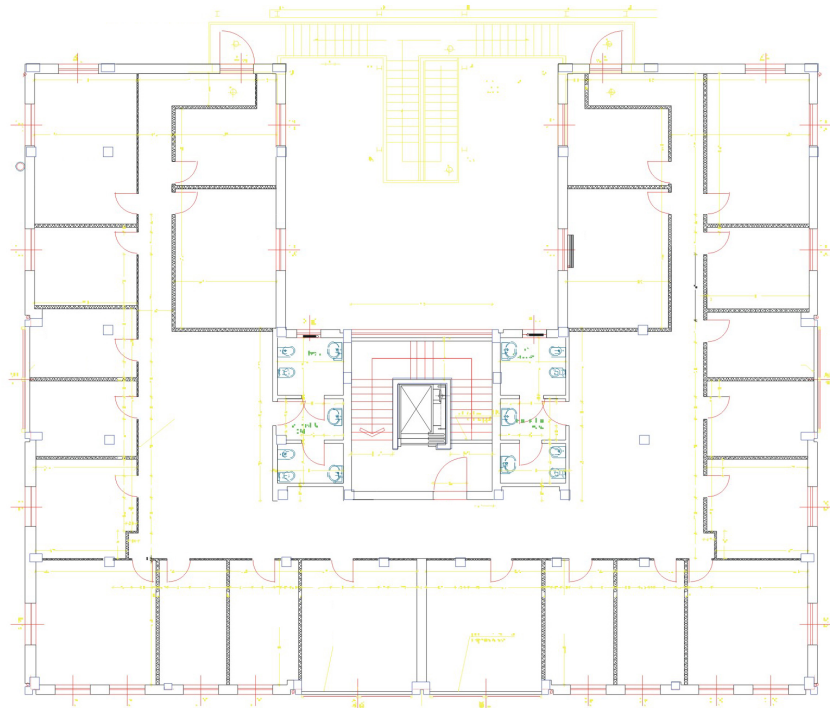
Una volta settata al meglio la rete sarà possibile confrontare con altre tecniche le prestazioni di questo tipo di localizzazione (vedi sez 6.3).

6.1 Use case: Edificio universitario

Per questi test si utilizzerà l'edificio universitario della seconda facoltà di Ingegneria di Bologna, con sede a Cesena, via Venezia n° 52. Questo edificio è composto da tre piani, nei primi due sono presenti laboratori e grandi stanze adibite a ricerca, e per questo motivo la localizzazione all'interno di esse sarà meno attendibile. Infatti i valori di potenza che di per se hanno una aleatorietà piuttosto alta in ampie stanze sono molto simili da punto a punto a causa di fading e ostacoli che possono variare (si pensi alle persone) con maggiore probabilità.

Il secondo piano dell'edificio invece è composto da piccoli studi, nei quali risiedono i professori, questo piano risulta ottimo per effettuare localizzazione, infatti i numerosi muri che separano l'ambiente permetteranno una maggiore differenziazione fra i pattern rilevati e perciò una maggiore probabilità di successo. Nella fig 6.1 ne viene mostrata la piantina.

Figura 6.1: Pianta secondo piano dell'edificio in via Venezia



Gli access point di questo edificio sono disposti al fine di dare massima

copertura nelle aree di interesse, perciò dove personale e studenti sono maggiormente interessati ad accedere alla rete internet. Se ne trovano quindi 4 al secondo piano, 2 al primo e 1 al piano terra.

6.2 Analisi del funzionamento della rete

Questa sezione cercherà di mostrare i risultati di localizzazioni effettuate su ognuno dei tre piano dell'edificio. In seguito ai risultati saranno perfezionati i parametri della rete e avverrà uno studio su come apportare eventuali modifiche all'algoritmo. L'ultimo test rappresenterà una sorte di dimostrazione finale dei risultati ottenuti dall'applicazione apportando le poche variazioni che il tempo a disposizione permetterà di attuare.

6.2.1 Primo Test: Piano terra dell'edificio

Prelevazione pattern e addestramento

In fig 6.2 sono riportati i punti di prelievo delle potenze. La tecnica usata è quella già citata, con prelievo verso i quattro lati per evitare interferenza del prelevatore, mediando successivamente tutti i valori. Sono state effettuate 100 rilevazioni per lato con un intertempo di 100 ms fra un prelievo e il successivo per permettere all'hardware di raccogliere effettivamente valori non ripetitivi.

Come si può notare si è cercato di prelevare pattern vicini ma separati da muri per determinare se effettivamente questi potessero variare significativamente i valori per un corretto impiego della rete. Nella tabella 6.1 si riportano tutti valori medi rilevati per ogni access point. Per gli access point non rilevati in certi punti, ma comunque segnalati dal sistema si è deciso di assegnare un valore significativamente basso di potenza pari a -100 dBm.

Attraverso i valori della tabella sono state calcolate in seguito le distanze euclidee fra i punti rilevati. Sono emersi valori minimi fra punti dello stesso cluster, e significativi fra punti di cluster diversi. Era quello che ci si aspettava dalla presenza di muri. Ovviamente nella stanza centrale, di enormi dimensioni, i valori hanno un andamento più casuale con uno spettro più ampio di potenze. Per questo risulta conveniente avere spazi piccoli, per evitare di incappare in situazioni di incertezza data la grossa varietà di valori,

Tabella 6.1: Potenze rilevate al primo terra

Cl.	Punto	AP1	AP2	AP3	AP4	AP5	AP6	AP7
1	1	-84.79	-89.50	-87.06	-89.11	-87.05	-89.25	-100
1	2	-84.54	-88.78	-88.60	-89.80	-86.52	-90.0	-100
1	3	-85.98	-90.80	-89.13	-90.70	-87.24	-90.0	-100
1	4	-88.66	-92.01	-89.04	-90.86	-88.29	-91.0	-100
1	5	-87.66	-91.84	-88.51	-90.13	-89.08	-91.0	-100
1	6	-87.89	-89.81	-87.72	-90.07	-88.66	-92.00	-100
1	7	-85.92	-90.30	-86.87	-87.60	-88.69	-90.07	-100
1	8	-87.21	-90.41	-88.61	-89.91	-89.68	-91.0	-100
1	9	-89.46	-90.21	-88.19	-90.13	-88.97	-91.72	-100
1	10	-87.85	-89.25	-88.23	-88.57	-86.41	-91.58	-100
1	11	-85.75	-88.43	-88.37	-89.93	-86.37	-91.0	-100
2	1	-81.95	-86.21	-85.56	-87.20	-87.32	-90.84	-100
2	2	-80.96	-86.83	-86.78	-89.91	-84.55	-90.0	-100
2	3	-82.94	-88.66	-89.11	-89.90	-86.60	-90.0	-100
2	4	-86.33	-87.94	-88.07	-88.72	-88.77	-89.79	-100
2	5	-85.64	-90.15	-87.17	-87.25	-89.76	-89.0	-100
2	6	-86.57	-100.0	-87.79	-90.08	-100	-100	-100
2	7	-88.55	-100	-88.64	-88.34	-100	-100	-100
2	8	-87.58	-86.64	-89.06	-91.81	-24.0	-100	-100
2	9	-85.95	-87.47	-87.19	-89.40	-39.00	-100	-100
2	10	-87.80	-88.0	-87.85	-89.95	-39.00	-100	-100
2	11	-88.48	-88.0	-89.29	-90.82	-61.00	-100	-100
2	12	-89.80	-90.65	-91.67	-91.0	-70.15	-100	-100
2	13	-88.51	-90.31	-90.68	-90.50	-86.31	-100	-100
3	1	-79.19	-89.0	-89.25	-90.06	-100	-100	-100.0
3	2	-84.67	-89.0	-86.42	-88.29	-100	-100	-100.0
3	3	-85.04	-89.0	-87.87	-89.99	-100	-100	-88.0
3	4	-84.56	-88.67	-87.15	-88.24	-100	-100	-88.0
3	5	-86.46	-88.0	-85.25	-88.74	-100	-100	-88.0
3	6	-86.94	-89.55	-85.51	-86.91	-100	-100	-88.0

Figura 6.2: Pianta dell'edificio al piano terra, ala nord



i quali potrebbero finire con l'essere vicini a valori di altri cluster.

Distanze spaziali piccole in cluster piccoli, che risulteranno le situazioni ricorrenti dell'ambiente in cui si effettueranno i test definitivi, hanno dato distanze euclidee di potenza che si aggirano sui 2-3 dBm. Precisamente con minimi di circa 1.5. In questa prima fase, si è voluto creare una rete evitando unificazione di pattern, rilevando punti a distanze significative (maggiore di due metri), ovvero ove ogni neurone nascosto gestisse un solo pattern. Per fare ciò si è deciso di fornire raggi con dimensioni dai 0.4 agli 1.3 dBm, intuendo dalle rilevazioni che 0.4 fosse la distanza minima che si può avere da una persona ad un'altra fisicamente vicine. Come raggio iniziale si è fissato un intermedio 0.8.

Ovviamente con 30 pattern di addestramento e questi parametri sono sorti 30 neuroni nascosti, ognuno dei quali agganciati ad uno dei tre cluster.

Il parametro learning rate è stato fissato ad un valore molto molto basso per scoraggiare un neurone di spostarsi di molto in caso di inserimento di un nuovo pattern nella sua area.

Il parametro di soglia di esclusione cluster è stato ridotto a 0 per portare

tutti i cluster a gareggiare nel riconoscimento di un pattern da localizzare.

Localizzazione

In seguito all'addestramento da file della rete di 30 pattern sono stati effettuati le prove di localizzazione. In realtà fra addestramento e localizzazione sono passati alcuni giorni e l'ambiente è stato leggermente cambiato (aggiunta mobili, spostamento tavoli, aumento di presenza di personale).

Sono sorti subito alcuni problemi.

Si è riscontrato che grazie alla funzione di attivazione dell'ultimo strato, nella quale veniva effettuata la somma delle uscite dei propri neuroni nascosti, i cluster con più neuroni nascosti erano avvantaggiati nella gara della localizzazione. Perciò si è rivisitato il progetto e si è deciso di trasformare la funzione di attivazione in un filtro che facesse passare soltanto il valore più alto dei neuroni nascosti, col quale il cluster proprietario gareggerà contro gli altri cluster.

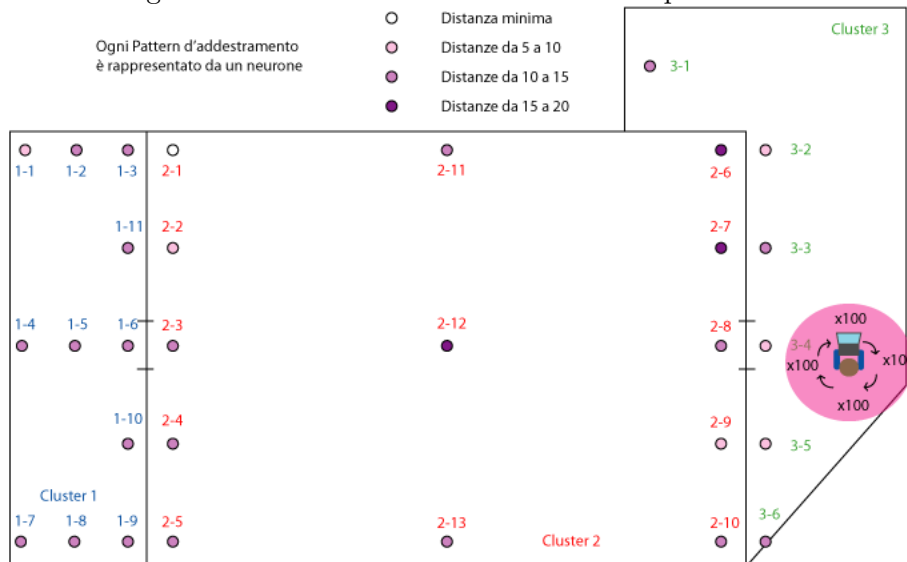
Inoltre si è notato che il calcolo della distanza euclidea non dava i frutti sperati. Infatti la distanza euclidea, che qui rappresenta la radice di una somma quadratica fra le differenze dei valori di potenza di un pattern da localizzare e i pesi di un neurone nascosto, risultava così fatta sconveniente perchè quando un pattern non localizzava alcuni access point di cui però i neuroni tengono traccia, veniva assegnato un valore ragionato (-100) uguale per tutti gli access point non rilevati.

A causa di ciò, alcuni pattern si avvicinavano di più a neuroni nascosti che avevano pesi pari a zero per non aver rilevato l'access point, essendo ognuno in questo test rappresentativo di un pattern d'addestramento in fase addestrativa, piuttosto che a neuroni nascosti effettivamente più vicini. Questa tecnica permetteva di avvicinare tanto pattern e neuroni che entrambi non avevano ricevuto l'AP. Perciò in questa fase si è deciso di trasformare il calcolo della distanze euclidea, sommando solo i quadrati delle differenze effettivamente calcolabili, ovvero di quei valori che il pattern di localizzazione ha rilevato.

Il primo rilevamento è stato effettuato nel cluster 3, girando sui quattro

lati, con 100 misure per lato come si può vedere in figura 6.3. Purtroppo ha dato esito negativo.

Figura 6.3: Primo test di localizzazione al piano terra



Il secondo rilevamento è stato fatto nel cluster 1 sempre con 100 rilevazioni per lato e l'esito è stato positivo (fig 6.4).

Siccome un utente per essere localizzato non potrà attendere molto, e soprattutto non potrà girare su se stesso, il terzo test è stato effettuato verso un solo lato, con 100 misurazioni.

Da questi risultati si è evidenziato che probabilmente la posizione dell'utente ha condizionato una vittoria del punto 1-7 come nel secondo test, infatti i punti nella direzione di prelievo come si può notare da fig 6.5 sono avvantaggiati rispetto agli altri. Il risultato è stato negativo ma considerando la presenza della porta l'errore è stato minimo.

Il quarto test (6.6) è stato effettuato nel cluster centrale ancora una volta senza ruotare su se stessi data l'improbabile disponibilità di un utente a fare ciò. Le misurazioni sono state sempre 100.

Avendo il quarto test dato esito negativo si è provato un cambio di lato visto che sembra incidere molto nel risultato il lato del prelievo.

Infatti nel quinto test è stata provata la direzione verso il terzo cluster sempre con 100 misurazioni.

Come si nota il risultato non è variato. Questo lo si può imputare al

Figura 6.4: Secondo test di localizzazione al piano terra

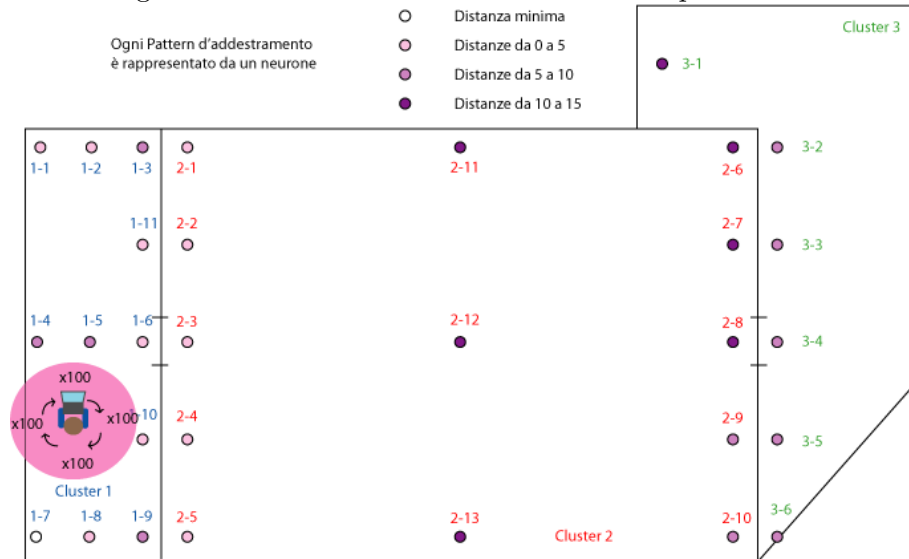


Figura 6.5: Terzo test di localizzazione al piano terra



Figura 6.6: Quarto test di localizzazione al piano terra

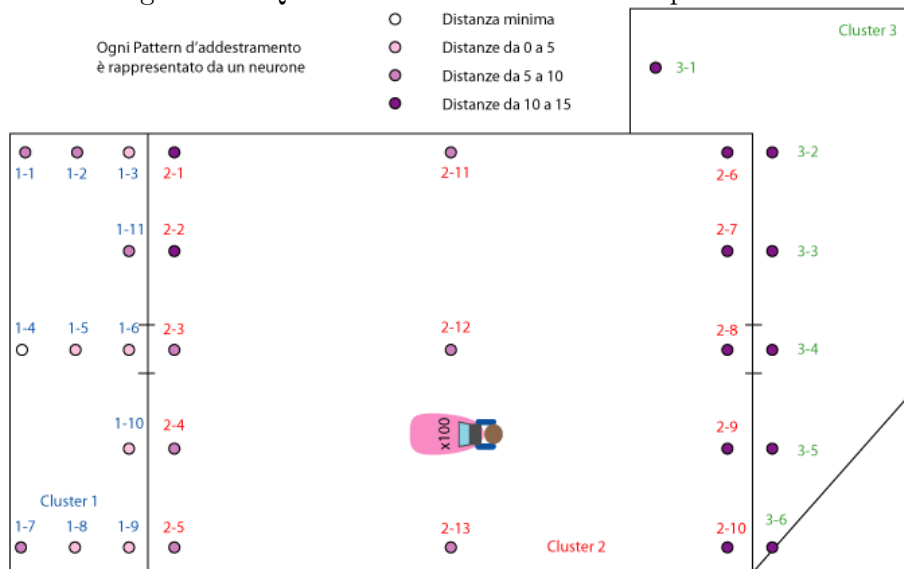
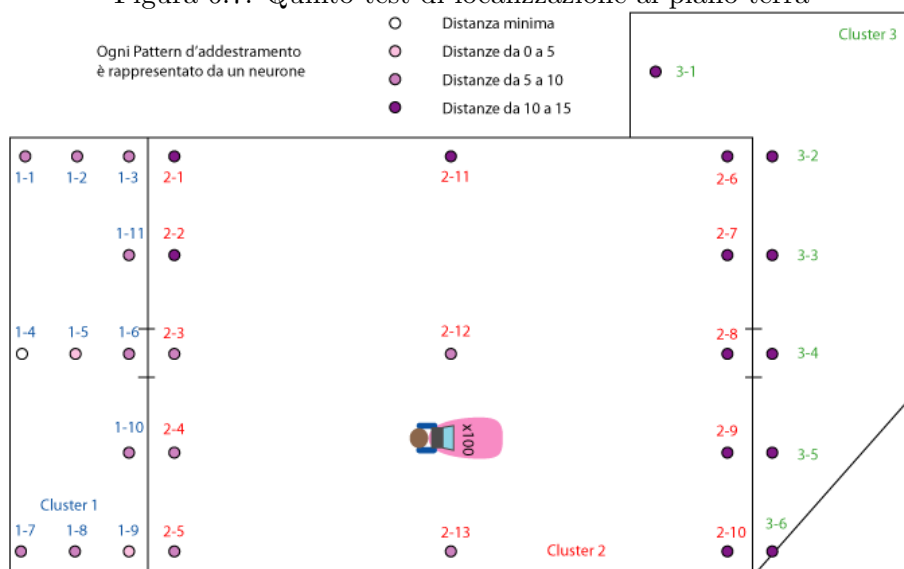


Figura 6.7: Quinto test di localizzazione al piano terra



fatto che il cluster dove è avvenuta la richiesta è di grandi dimensioni, e le potenze possono essere maggiormente soggette a disturbi.

6.2.2 Secondo Test: Primo piano dell'edificio

Prelevazione pattern e addestramento

Questo secondo ambiente risulta già più simile all'ambiente dove verrà impiegato il sistema finale. Nella mappa (fig 6.8) si può contraddistinguere un corridoio e stanze piccole anche se di forma molto variabile. I punti riportati sono quelli dove sono state fatte rilevazioni distinte per cluster. Il problema maggiore di questo ambiente è risultato la divisione fra cluster, infatti molti ambienti sono divisi solamente da alte scaffalature che non raggiungono neanche il soffitto.

Anche in questo caso le potenze non riportate in certi punti sono state catalogate come potenze a -100 dBm, un valore significativo che permetterà all'algoritmo della rete di utilizzarne il significato.

Si riportano le potenze in tabella (tab 6.2).

Verranno costruite due reti per denotare l'effetto nella localizzazione, nella prima ogni neurone rappresenta un pattern d'addestramento e nella seconda si lascia la possibilità di inglobare più pattern in un neurone nascosto. Per fare ciò definiamo parametri differenti.

Per prima cosa viene definito un nuovo modo di calcolare le distanze. Infatti ragionamenti logici hanno portato a dedurre per ora che le distanze, sia fra pattern di addestramento (servono per avere un'idea di quali sono le dimensioni virtuali dell'ambiente) che fra pattern e pesi del neurone (calcolata per determinare il neurone più vicino), necessitano l'esclusione dei valori di potenza non rilevati in uno o nell'altro vettore, inputs del calcolo. Infatti se per uno dei due vettori non si ha il valore di potenza, si proseguirà in un calcolo della distanza euclidea che non tiene conto di quella componente. Così da escludere errori apportati dal mancato rilevamento di alcuni access point non raggiungibili.

A questo punto le distanze fra i pattern di addestramento hanno denotato valori che vanno dai 4 ai 40 dBm circa. Perciò per una rete del primo tipo si decidono raggio iniziale pari a 1, minimo 0.4 e massimo 2.5.

Figura 6.8: Pianta dell'edificio al primo piano, ala sud

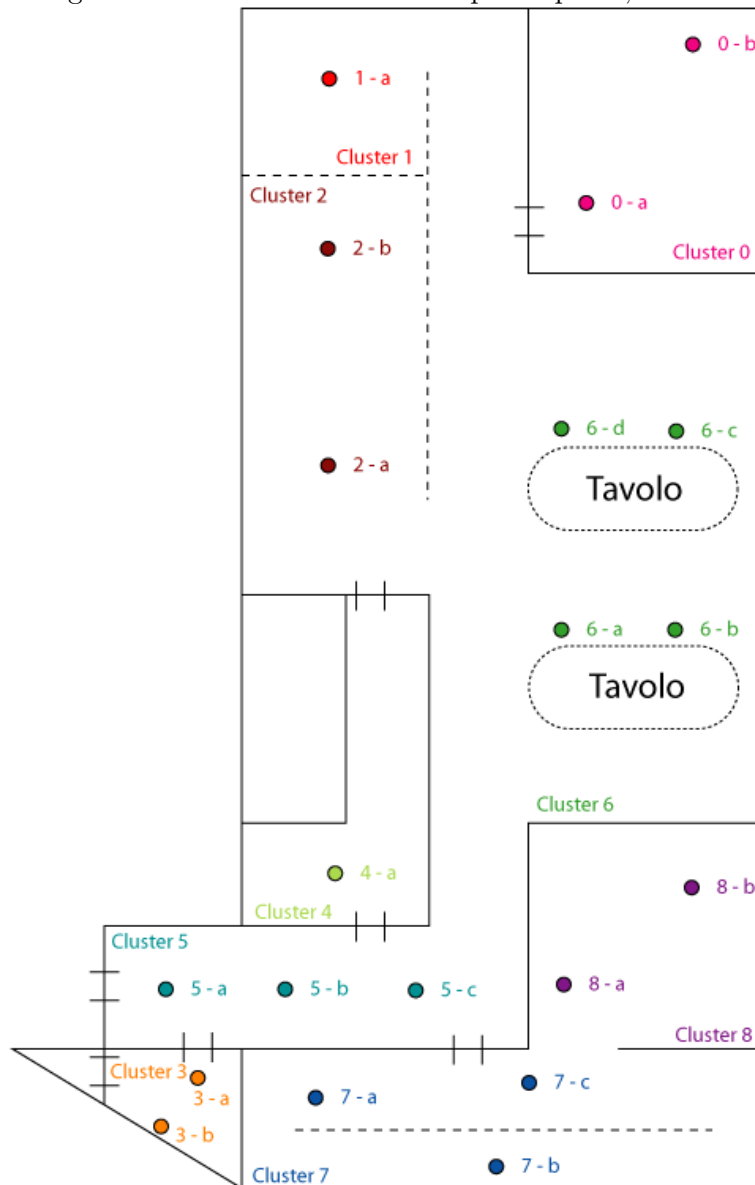


Tabella 6.2: Potenze rilevate al primo piano

Cl.	Pun.	AP1	AP2	AP3	AP4	AP5	AP6	AP7	AP8
0	a	-63.06	-89.01	-87.87	-87.22	-100.0	-100.0	-100.0	-100.0
0	b	-67.97	-86.33	-83.46	-82.33	-72.00	-100.0	-100.0	-100.0
1	a	-66.19	-88.64	-80.08	-85.01	-82.11	-75.20	-100.0	-100.0
2	a	-54.94	-85.94	-85.18	-82.88	-79.22	-77.99	-100.0	-100.0
2	b	-63.32	-87.18	-83.05	-84.57	-80.95	-76.25	-100.0	-100.0
3	a	-76.61	-87.00	-87.00	-83.22	-84.0	-71.0	-69.33	-100.0
3	b	-76.69	-87.00	-87.00	-79.30	-84.0	-71.0	-61.35	-100.0
4	a	-74.63	-81.84	-85.92	-82.51	-84.0	-71.0	-72.47	-100.0
5	a	-75.06	-85.46	-85.0	-84.99	-85.86	-85.99	-86.14	-70.0
5	b	-76.51	-84.48	-85.0	-83.31	-87.84	-85.99	-87.59	-70.0
5	c	-68.76	-87.40	-85.0	-85.23	-91.17	-85.99	-86.84	-70.0
6	a	-50.83	-86.40	-79.21	-79.58	-87.86	-85.99	-87.82	-70.0
6	b	-56.08	-86.35	-83.60	-77.62	-86.87	-85.99	-88.0	-70.0
6	c	-58.13	-88.59	-88.36	-77.99	-85.53	-85.99	-88.0	-70.0
6	d	-54.03	-88.53	-84.13	-80.68	-85.0	-79.96	-88.0	-70.0
7	a	-71.69	-88.41	-72.99	-82.08	-78.31	-74.99	-80.99	-70.0
7	b	-66.10	-71.48	-72.99	-80.28	-78.36	-79.21	-80.99	-70.0
7	c	-75.37	-87.62	-74.92	-90.02	-86.75	-82.65	-81.14	-70.0
8	a	-63.76	-90.86	-80.0	-69.76	-83.06	-87.00	-83.69	-70.0
8	b	-64.27	-87.54	-79.25	-70.56	-85.61	-87.00	-88.0	-70.0

Siccome le distanze fra cluster diversi superano quasi sempre i 10 dBm, per la seconda rete i parametri saranno di raggio iniziale pari a 5, minimo 0.4 (che risulta sempre la distanza minima virtuale di potenza fra due persone vicine) e massimo 10.

Il learning rate viene fissato ancora a 0.03 per entrambe le reti in modo da non spostare troppo un neurone e soglia 0 per fare gareggiare tutti i cluster.

Localizzazione

Si ricorda che nella prima fase verrà utilizzata una rete con neuroni nascosti rappresentanti ciascuno un pattern.

Tutti i punti di localizzazione hanno la caratteristica di essere stati prelevati senza rotazione con 100 misurazioni mediate. Non è trascorsa più di un'ora fra il prelievo dei pattern d'addestramento e quelli di localizzazione al fine di non avere un notevole cambiamento dell'ambiente.

Nelle figure che seguono (6.9, 6.10, 6.11, 6.12, 6.13, 6.14, 6.15, 6.16) sono chiaramente visibili i cluster vincitori, ovvero quelli con il neurone a distanza minima e si riportano anche le distanze dagli altri neuroni, ricordiamo che questi sono fisicamente rappresentabile dal fatto che ogni pattern prelevato per l'addestramento ha fornito rispettivamente ad ognuno i pesi.

Come si può notare i risultati non sono stati molto soddisfacenti, nonostante il calcolo della distanza euclidea sia cambiato. Le variabili in gioco sono diverse, sicuramente la più incidente è l'aleatorietà dei valori di potenza che mediati poche volte non forniscono un buon pattern di localizzazione.

Un altro punto da considerare è certamente la direzione, ovvero l'impossibilità di comandare l'utente di ruotare su se stesso per essere localizzato. Si nota infatti che la maggior parte delle volte vengono favoriti i punti che si trovano in direzione dell'utente. C'è da considerare che ogni antenna potrebbe avere una sua direzione non determinabile dall'esterno e perciò non sarà possibile predefinire in che direzione l'utente andrà a mettersi per essere localizzato.

La seconda fase consiste nell'utilizzare la rete addestrata con parametri

Figura 6.9: Primo test di localizzazione al primo piano

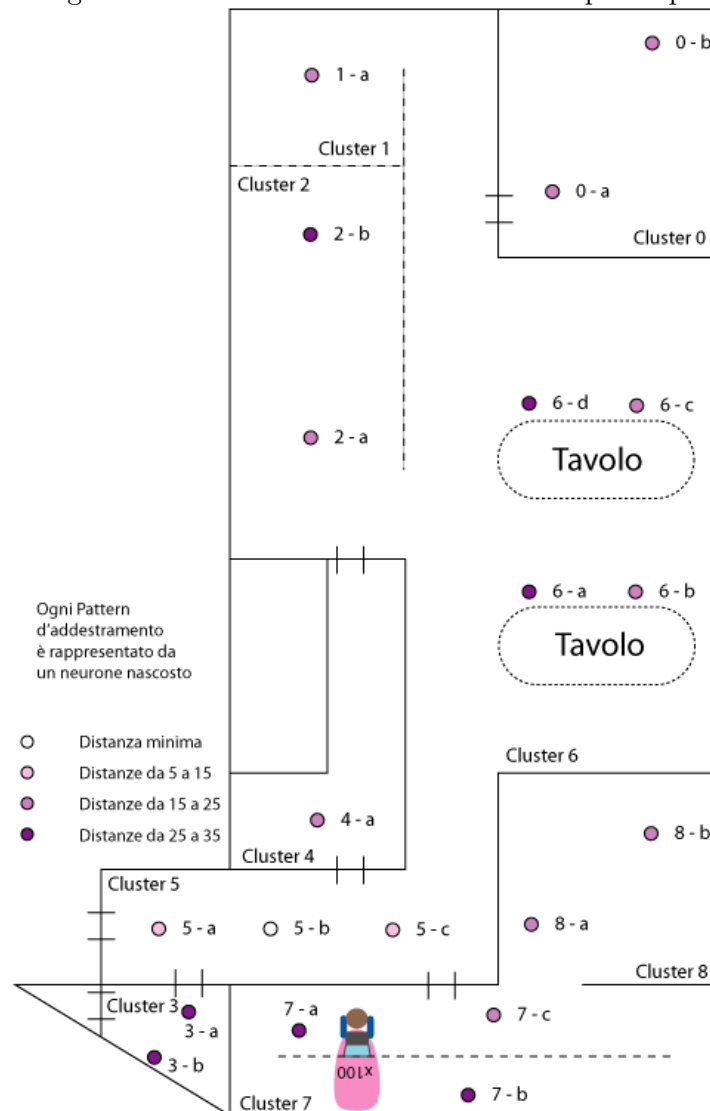


Figura 6.10: Secondo test di localizzazione al primo piano

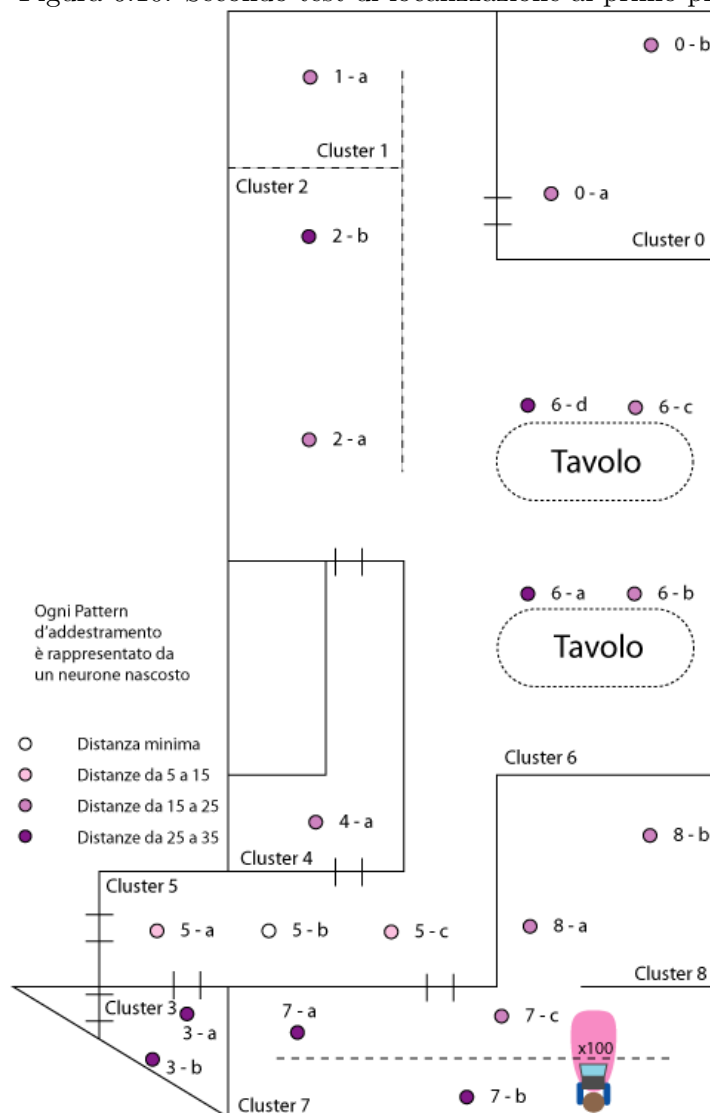


Figura 6.11: Terzo test di localizzazione al primo piano

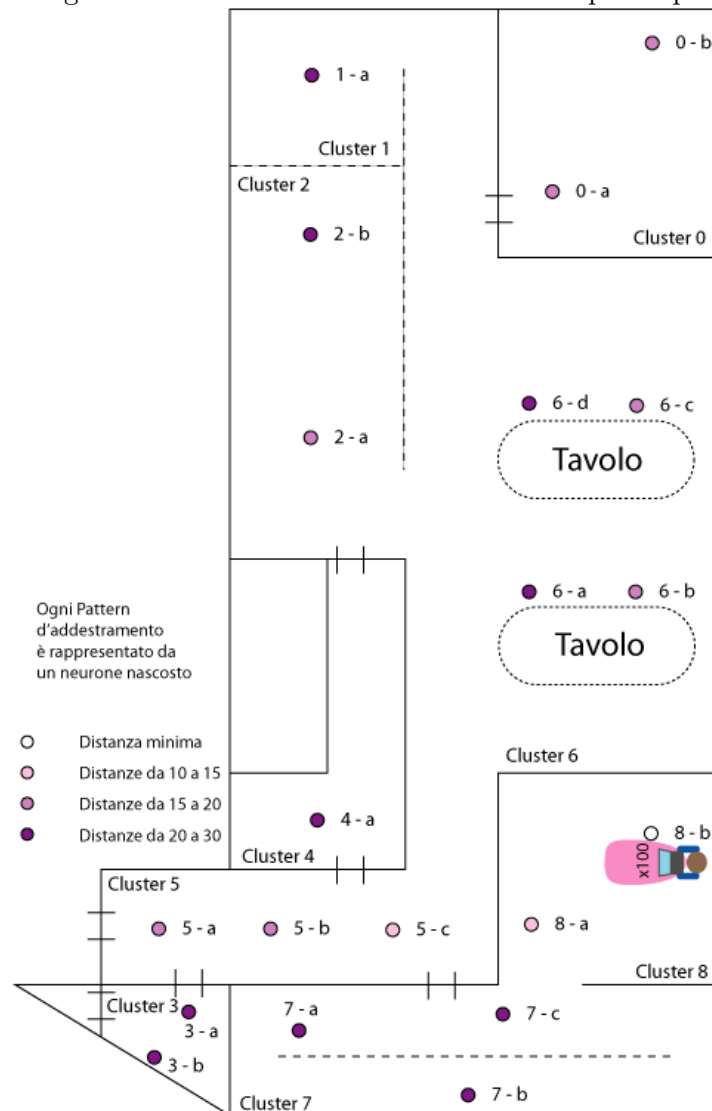


Figura 6.12: Quarto test di localizzazione al primo piano

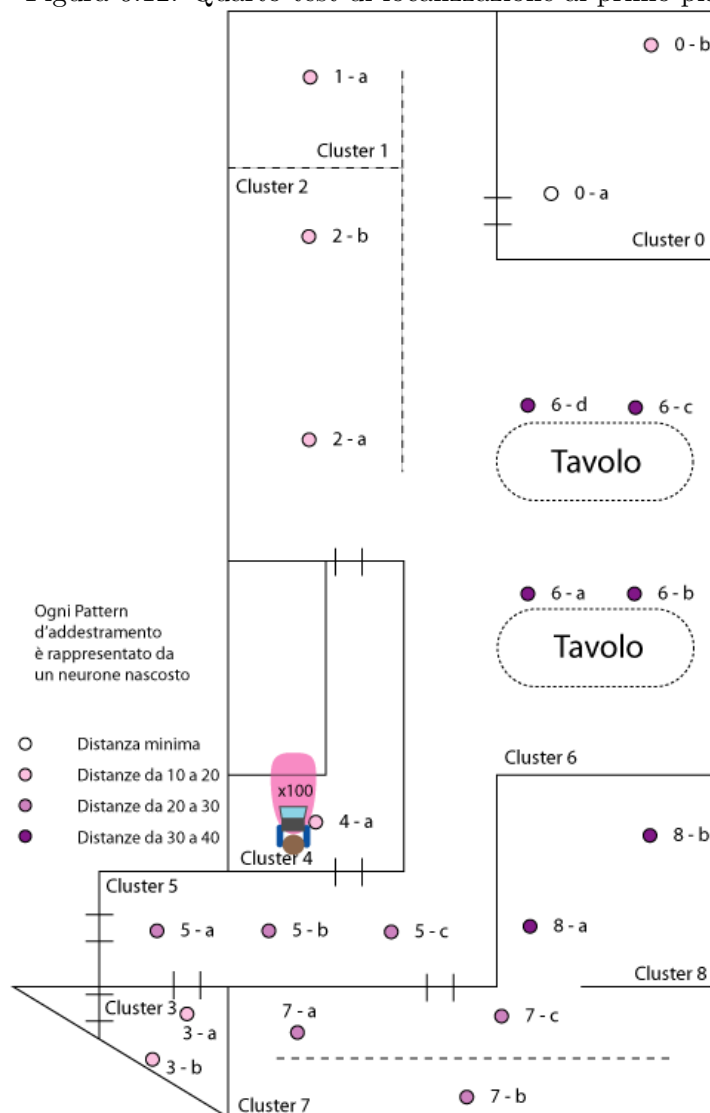


Figura 6.13: Quinto test di localizzazione al primo piano

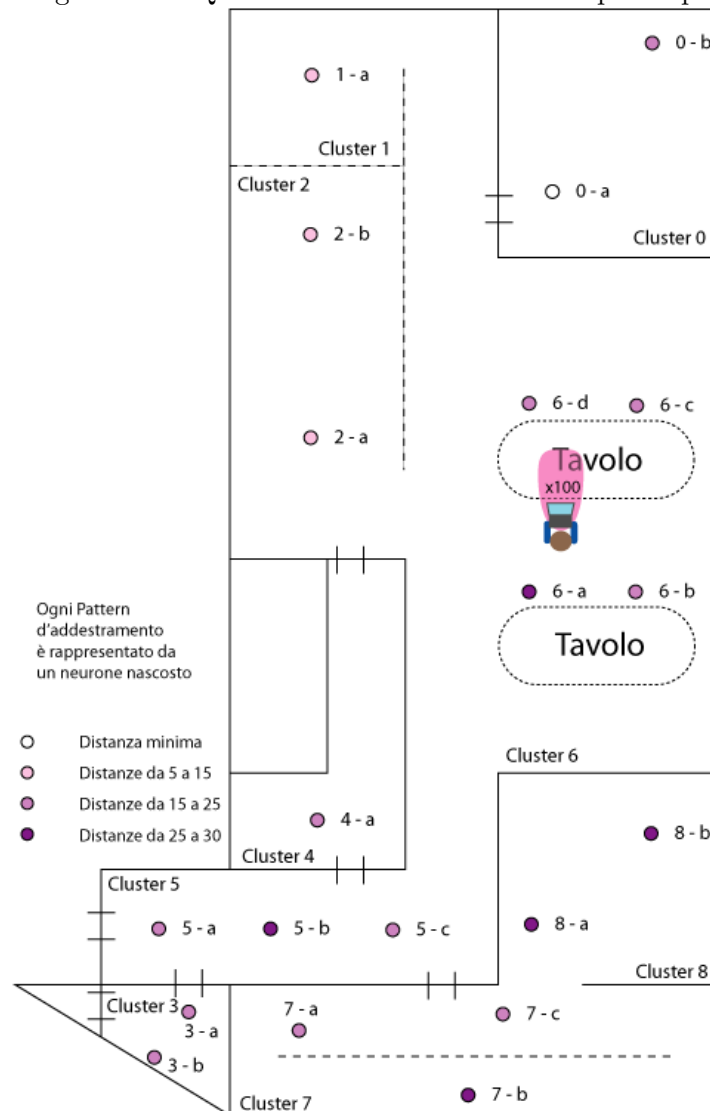


Figura 6.14: Sesto test di localizzazione al primo piano

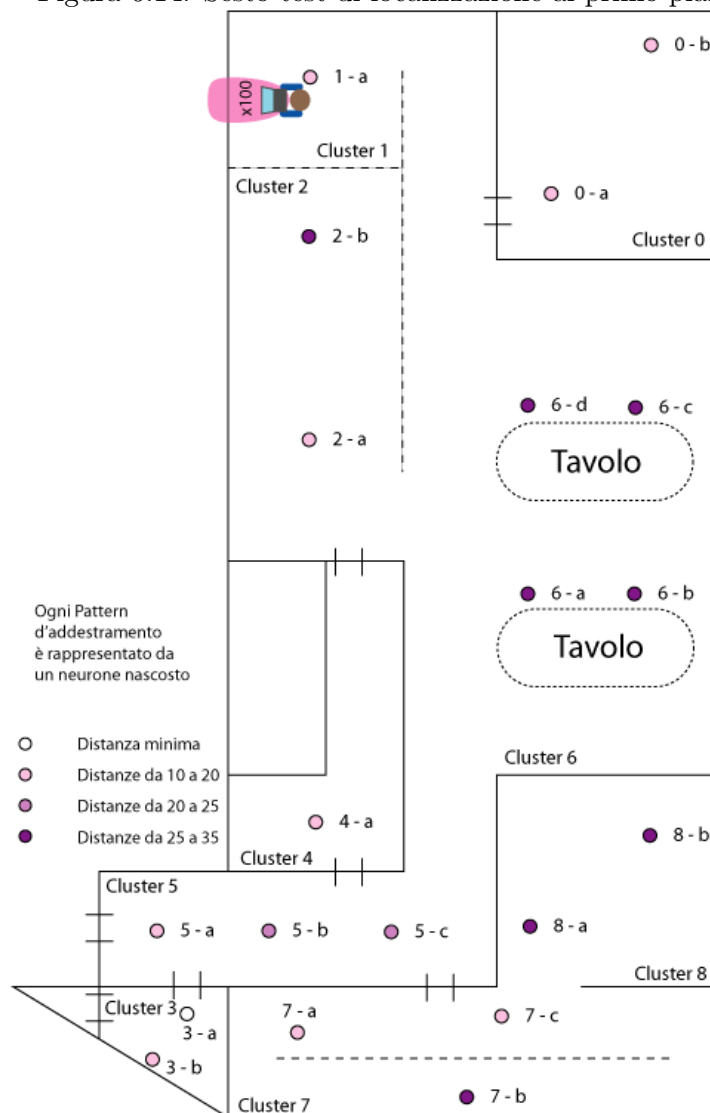


Figura 6.15: Settimo test di localizzazione al primo piano

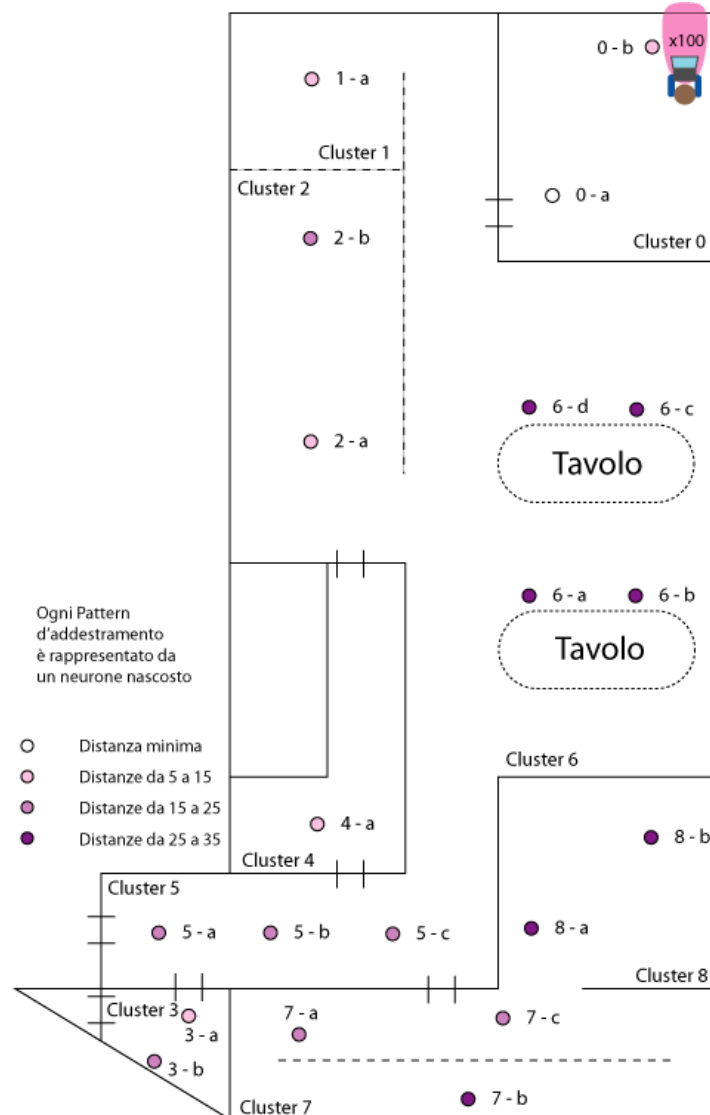
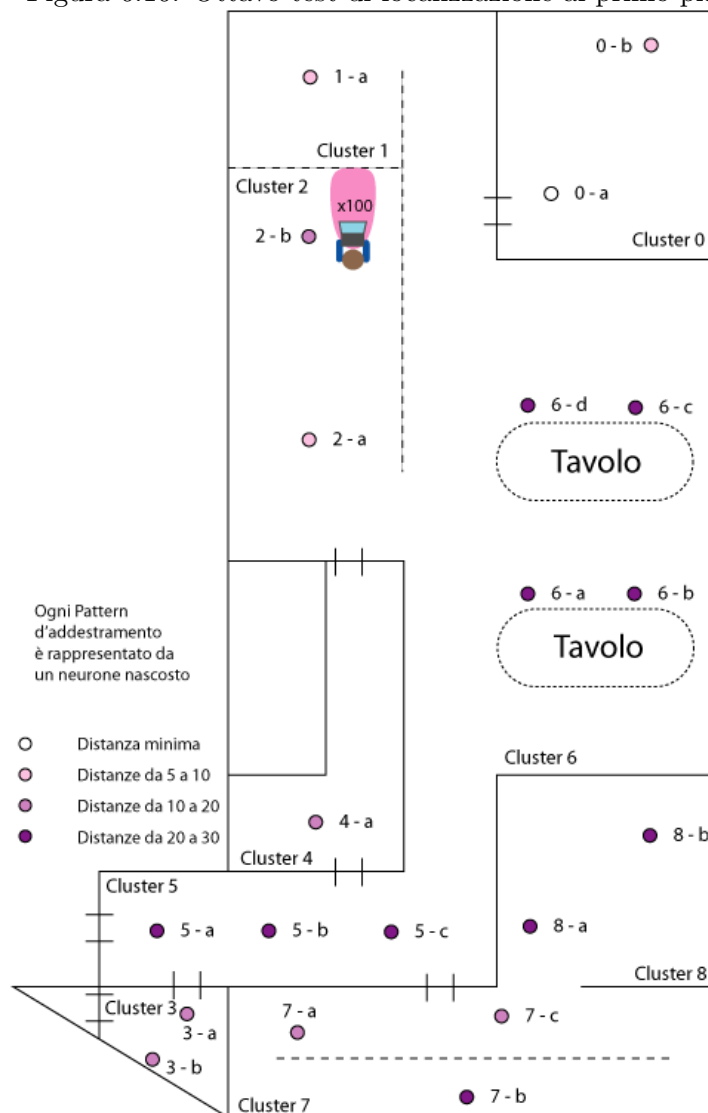


Figura 6.16: Ottavo test di localizzazione al primo piano



diversi. Infatti con i parametri suddetti accadrà che alcuni pattern di un cluster andranno a fondersi in un solo neurone nascosto.

I risultati sono stati i seguenti inserendo gli stessi pattern di localizzazione della precedente fase.

A parte i cluster con valori più distanti (6 e 7), quasi tutti gli altri cluster hanno unificato i pattern all'interno di un unico neurone nascosto. Questa cosa rischia l'intersecamento di zone gestite dai cluster, ma non sarà questo un problema, quanto più rischia di centrare i neuroni in un punto unico del cluster, probabilmente quello del primo pattern avendo un valore di learning rate molto basso.

Per il primo punto di localizzazione si è ottenuto il cluster 5 invece che il 7 (il suo neurone nascosto più vicino è ancora molto distante).

Il secondo punto ha dato ancora vittorioso il cluster 5 al posto del 7, come prima. Lo si può imputare al fatto che l'unica barriera fra i due cluster era un pannello di vetro che probabilmente non forniva un buon isolante per avere valori di potenza molto diversi.

Il terzo punto ha dato vincitore il cluster 5 al posto dell'8 anche se l'unico neurone nascosto del cluster del pattern reale dista pochissimo di più dal vincitore. Probabilmente è dovuto al fatto che il cluster 8 unificando ha spostato la sua area di competenza che prima non unificando lo rendeva vincitore in questa prova.

Il quarto punto ha dato vincitore il cluster 2. Meglio che il cluster 0 della precedente prova ma ancora non basta. Positivo è che il terzo neurone più vicino è proprio quello del cluster 4.

Il quinto punto ha dato vincitore ancora il cluster 2. Risulta ben lontana la possibilità di appartenere al cluster 6, come dovrebbe essere. Questo problema potrebbe venire risolto con una modifica della funzione euclidea, ma se ne parlerà in seguito.

Il sesto punto fa vincere il cluster 3 come nella precedente sperimentazione; una buona notizia risulta comunque che il neurone del cluster 1, si trova al secondo posto nella lista delle distanze minime.

Il settimo punto da completamente perdente il cluster 0 probabilmente dovuto all'unificazione dei suoi due pattern d'addestramento.

L'ultimo punto converge nel cluster giusto, il 2. Probabilmente per lo stesso identico motivo del precedente errore. Infatti se nella prima fase

aveva vinto il cluster 0, in questa perde per il proprio spostamento dato dall'unificazione.

Si può concludere che l'effetto unificativo varia da cluster a cluster, a seconda della propria morfologia.

6.2.3 Soluzioni proposte

In seguito ai risultati ottenuti si potrebbe a lungo discutere sulle numerose modifiche da apportare alla rete. Si decide di soffermarsi sulle più importanti.

Inserimento di memoria nella rete

Questa sarebbe senza meno la soluzione che più darebbe contributo al sistema. Infatti si parte dal presupposto che un utente non possa spostarsi all'interno della facoltà ad una elevata velocità, perciò si potrebbe influenzare la scelta di una posizione escludendo le posizioni talmente lontane da non potere essere effettivamente quelle corrette.

Un'idea sarebbe quella di dare un coefficiente ad ogni neurone che permetta a quelli vicini al neurone vincitore precedente (utilizzando la distanza euclidea) di essere avvantaggiati nella gara. Il problema risulta la struttura del progetto sino a qui elaborato. Infatti questa rete neuronale non contiene le caratteristiche adatte per mantenere la storia passata delle gare al suo interno. Bisognerebbe modificare sostanzialmente la struttura o fornire la rete di artifici quali l'inserimento nel pattern di localizzazione delle posizioni passate dell'utente. Siccome il sistema a servizi vuole disporre delle reti neurali come strumenti istantanei e utilizzabili anche per più utenti contemporaneamente, dedicare una sola rete per un utente trasformerebbe troppo il sistema a pochi giorni dalla conclusione del progetto. Tutto ciò si rimanda come idea per evoluzioni future.

Funzione di attivazione dei neuroni di uscita

L'idea di mutare la funzione d'attivazione dei neuroni d'uscita ha un po' decentinato la presenza della rete. Infatti, questo passo avvicina la rete neuronale sempre più alla tecnica RADAR, che consiste solamente in un confronto fra pattern di localizzazione e vettori della mappa radio.

Si potrebbe pensare in futuro di riprendere in mano l'idea di una somma pesata come funzione di attivazione dell'ultimo strato, rendendola però funzionale attraverso una normalizzazione di essa, dipendente dal numero di neuroni nascosti facenti parte del cluster. In questo modo non si avrebbe più il vantaggio dei cluster con più prelievi, ma con una semplice media sul numero di neuroni nascosti di ognuno si potrebbero avere miglioramenti sulla discriminazione dei cluster stessi.

Distanza euclidea

La distanza euclidea non è in alcun modo modificabile come strumento per valutare la vicinanza fra pattern di potenza a causa delle caratteristiche dei segnali in ambiente indoor. Infatti essa nasce dalla ottimalità del criterio MAP per determinarne la stima. Ciò è sostenibile dal fatto che la densità di probabilità in un ambiente indoor dovrebbe avere l'andamento di una gaussiana.

Se non fosse che in seguito a studi effettuati nell'edificio [10] si è dedotto che il grafico della densità di potenze di qualsiasi access point determinato in un punto non corrisponde proprio ad una funzione gaussiana ma ci assomiglia. Ciò scardina il concetto dell'ottimalità del criterio MAP e lascia porte aperte per l'introduzione di nuove distanze utilizzabili per i confronti.

Inoltre, anche se la curva fosse gaussiana, l'unificazione di pattern in una rete neurale all'interno di un unico neurone nascosto trasforma la densità di probabilità in qualcosa che sicuramente non corrisponderà ad una curva gaussiana. Questo potrebbe essere un altro argomento di studi futuri.

Aumento delle rilevazioni per lato

Sino ad ora sono state effettuate rilevazioni ogni 4 m^2 circa. Quindi in stanze piuttosto piccole le rilevazioni sono risultate minime, non più di 2. L'aleatorietà delle potenze non aiuta ciò.

Infatti anche se la rete nel suo addestramento non accomuna più rilevamenti in un neurone nascosto (non fa cioè unificazione o spostamento di quest'ultimo), quelli che rappresentano un cluster rischiano veramente di essere esigui. Si punta quindi ad aumentare il loro numero facendo partecipare

più neuroni alla gara e aumentando la probabilità che il vincitore si trovi effettivamente vicino.

Valutazione di AP rilevati

Il problema nasce in aree in cui non ci si trova coperti dallo stesso set di access point comune per la maggior parte dei cluster. Infatti non di rado accade di rilevare punti in cui il set di access point risulta costituito da più componenti rispetto la maggior parte degli altri punti. Questo dovrebbe determinare un aiuto nella fase di localizzazione.

Infatti rilevare un access point in più potrebbe essere la discriminante per escludere dalla localizzazione un insieme di cluster dove risulta impossibile rilevarlo. Il problema nasce dall'aleatorietà del rilevamento. Potrebbe essere rilevato una prima volta e poi non più in seguito.

Una soluzione già adottata in altri sistemi risulta quella di escludere dalla gara delle misurazioni della mappa radio, nel nostro caso dei neuroni nascosti, che non hanno lo stesso set di AP rilevati del pattern da localizzare. Il rischio è quello di ritrovarsi con un set di AP rilevati del pattern da localizzare diverso dal neurone nascosto più idoneo per vincere la gara per colpa della variabilità con la quale si rilevano vengono rilevati gli access point.

Si pensa quindi, come ulteriore evoluzione del sistema, di fornire un vantaggio ai neuroni che hanno pesi per gli stessi access point che il pattern di localizzazione ha rilevato. In questo modo vinceranno con più facilità i neuroni con lo stesso set di potenze ma non si dimentica la possibilità di cambiamenti nel set di AP della rilevazione, lasciando comunque gareggiare ogni neurone nascosto.

Un'altra possibile soluzione, che verrà illustrata nella successiva sezione, sarà quella di considerare un sottoinsieme di potenze, ovvero quelle più elevate (si ricorda che potenze non rilevate assumono valori pari ad un minimo prestabilito).

Confronto con valori più potenti del pattern

Un'ulteriore caratteristica della rete che verrà facilmente implementata sarà quella di calcolare la distanza euclidea fra i pesi del neurone nascosto e quelli

del pattern di localizzazione solo con le componenti più potenti di quest'ultimo. In questo modo il numero di componenti confrontate attraverso la distanza non varierà da neurone nascosto all'altro ma sarà possibile scartare dal calcolo le componenti minori del set di AP del punto da localizzare.

Considerazione della direzione di rilevazione

In seguito a ragionamenti fatti per la non ottimalità delle soluzioni, ci si è chiesto se l'idea di mediare i quattro lati di rilevazione per un punto fosse corretta. Infatti il rischio, in caso di sostanziale differenza di valori pescati in un verso o in un altro, risulta essere quello di stravolgere i valori sino ad averne una media distante da tutti e quattro i vettori di potenza rilevati sui 4 lati predefiniti.

Una soluzione possibile potrebbe essere quella di distinguere le rilevazioni anche per lato e di fare localizzazione con individuazione di direzione dell'utente. Ovviamente questa è una ulteriore capacità che per ora non interessa; lo scopo principale di questa modifica risulterà quello di localizzare con più facilità un utente in un determinato cluster.

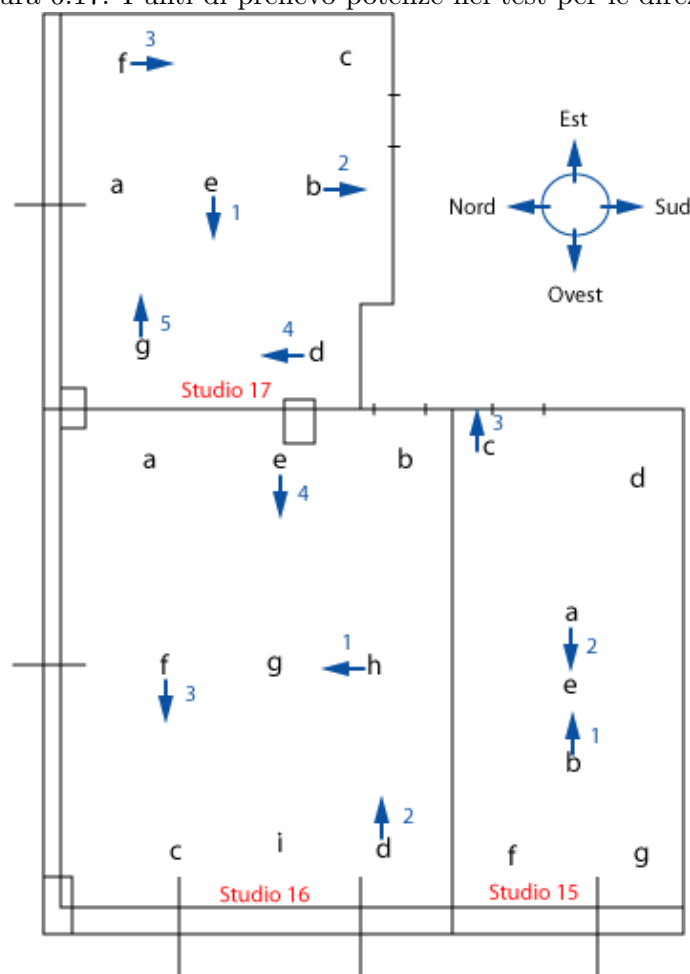
Per risolvere il dilemma di quale tecnica utilizzare si sceglie di effettuare un test. Questo test si svolgerà all'interno di tre stanze vicine, ognuna coperta da un numero consistente di rilievi. Le rilevazioni verranno distinte fra quelle effettuate mediando le misure per lato e quelle che mantengono entrambi i 4 vettori di potenza per ogni punto. Si avranno perciò due reti addestrate diversamente. Quella che in seguito alla fase di localizzazione sarà più efficiente determinerà il metodo di addestramento seguito.

La figura 6.17 mostra le tre stanze e i punti nelle quali verranno effettuate le rilevazioni.

Nel primo caso, come già citato, verranno prelevate le potenze mediando 100 misurazioni per lato di ogni punto. Nel secondo caso invece verranno effettuate sempre 100 misurazioni per lato, ma dopo aver effettuato la media per ognuno di loro verranno consegnati alla rete 4 pattern di addestramento, ognuno rappresentante una direzione cardinale. Quest'ultimo risulterà un addestramento quadruplicato che con parametri corretti della rete creerà un numero di neuroni nascosti quadruplicato rispetto il test precedente.

Nella fig 6.17 le freccette indicano i punti dove sono stati raccolti i valori

Figura 6.17: Punti di prelievo potenze nel test per le direzioni



di potenza per effettuare localizzazione e determinare il caso migliore. La direzione della freccia indica la direzione del prelievo. Questi prelievi sono stati effettuando mediando 20 misurazioni.

Si sceglie di non riportare tutte le potenze rilevate ma di fornire solo i risultati al fine di non creare eccessiva confusione.

In realtà i test effettuati saranno 6 per l'impiego di alcune variabili:

- Test1: Addestramento con media delle direzioni su tutti gli AP rilevati
- Test2: Addestramento senza media di direzione su tutti gli AP rilevati
- Test3: Addestramento con media delle direzioni solo con AP propri dell'edificio
- Test4: Addestramento senza media di direzione solo con AP propri dell'edificio
- Test5: Addestramento con media delle direzioni con i 4 AP più potenti sui 6 rilevati
- Test6: Addestramento senza media di direzione con i 4 AP più potenti sui 6 rilevati

Nella seguente tabella 6.3 vengono riportati i punti dove sono stati fatti i rilievi, molto vicini a punti per l'addestramento (identificabili attraverso questa vicinanza), e i risultati forniti dalla rete, ovvero il neurone nascosto vincitore. Siccome si è voluto addestrare la rete in modo che ogni neurone nascosto rappresentasse un pattern d'addestramento, il neurone nascosto vincitore verrà identificato con il pattern d'addestramento che lo ha creato. Semantica dei valori della tabella: $\langle\langle studio \rangle\rangle$: $\langle\langle punto \rangle\rangle$ [- $\langle\langle direzione \rangle\rangle$]

I seguenti test portano a fare numerose considerazioni.

La prima è quella che l'aumentato numero di pattern d'addestramento consegnato alla rete in confronto ai test eseguiti ai piani inferiori ha dato ottimi risultati. Infatti è raro che la rete sbaglia cluster.

Per di più la rete non sbaglia mai cluster se i valori sono mediati nelle potenze.

Non si nota gran differenza dal numero di access point considerato. Rimane comunque forte l'idea di non poter considerare in fase di addestramento access point non propri dell'edificio per la semplice motivazione che

Tabella 6.3: Risultati dei test sulle direzioni

Punto reale	Test1	Test2	Test3	Test4	Test5	Test6
15:b-E	15:f	16:b-S	15:f	16:b-S	15:f	16:b-S
15:a-S	15:e	16:b-O	15:e	16:b-O	15:e	16:b-O
15:c-E	15:e	15:b-N	15:e	15:b-N	15:a	15:b-N
16:h-N	16:e	16:i-E	16:e	16:i-E	16:e	17:b-N
16:d-E	16:g	15:f-O	16:g	15:f-O	16:g	15:f-O
16:f-O	16:c	16:i-O	16:c	16:i-O	16:c	16:i-O
16:e-O	16:b	16:g-O	16:b	16:g-O	16:b	17:a-O
17:e-O	17:d	17:g-S	17:d	17:g-S	17:d	17:c-S
17:b-S	17:e	17:e-E	17:e	17:e-E	17:e	17:e-E
17:f-S	17:e	17:b-O	17:e	17:b-O	17:e	17:f-S
17:d-N	17:e	17:e-E	17:e	17:e-E	17:e	17:a-O
17:g-E	17:b	17:d-O	17:b	17:d-O	17:b	17:g-S

qualsiasi loro modifica può non essere conosciuta e potrebbe avvenire una variazione dell'ambiente reale senza che la rete venga modificata di conseguenza. Diminuire il numero di access point considerati in base a quelli di potenza più elevata porta differenza soltanto per valori non mediati nelle direzioni. Aumentano gli errori di stanza ma aumenta anche l'esatta correttezza sino a livello di neurone nascosto di alcune localizzazioni.

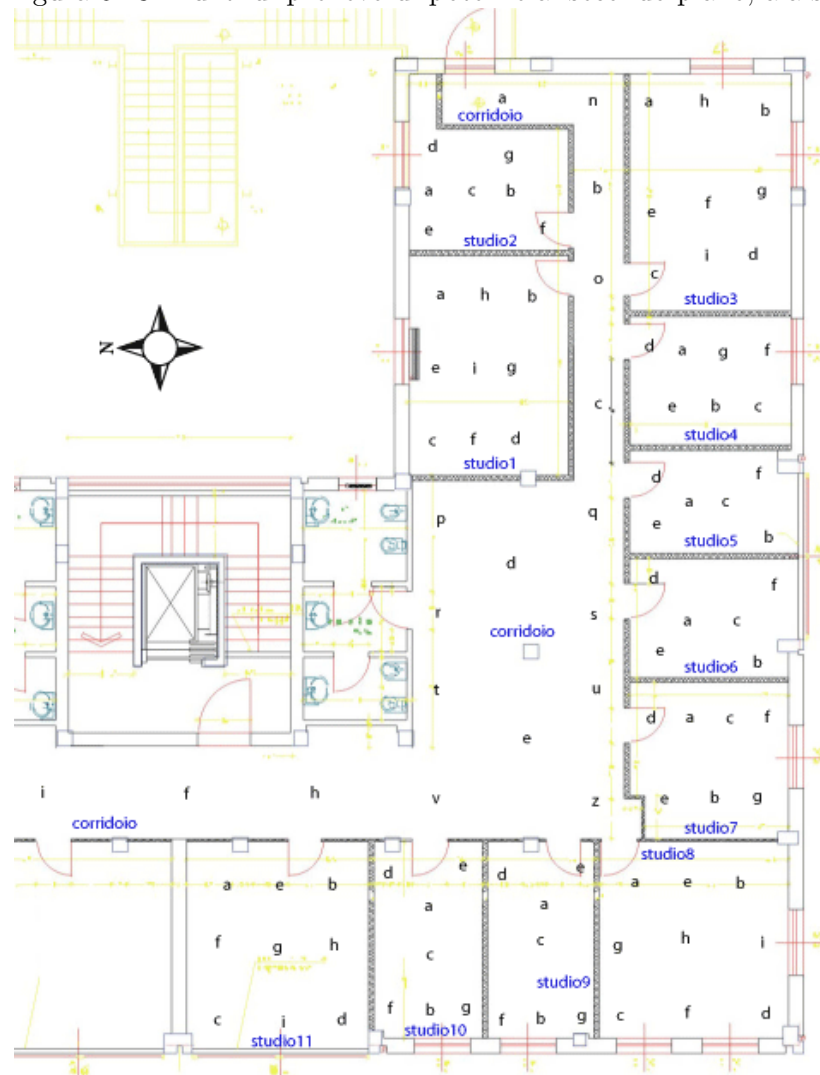
In seguito a questa esperienza, dato il fatto che la rete a questo stato dell'arte non fornisce la posizione esatta all'interno del cluster ma solamente la presenza all'interno di un cluster, si opta per continuare a mediare i valori sulle 4 direzioni ma sicuramente con un numero alto di rilievi per ogni cluster.

6.2.4 Terzo Test: Secondo piano dell'edificio

In seguito alle deduzioni fatte dai capitoli precedenti si è giunti a testare l'applicazione nell'ambiente in cui dovrebbe fornire i migliori risultati. A causa della considerazione di aumentare i punti di mappatura per ogni stanza si è optato per utilizzare solo metà del piano, visto lo stringersi dei tempi.

La figura 6.18 mostra i punti nei quali sono state prelevate le potenze per la costruzione della mappa radio.

Figura 6.18: Punti di prelievo di potenze al secondo piano, ala sud



Come nella maggior parte dei test precedenti si cercherà di minimizzare l'unificazione di pattern sotto un unico neurone nascosto fissando raggio e limiti molto piccoli. Infatti i parametri della rete varranno rispettivamente 0.8, 0.4, 1 (raggio iniziale, minimo e massimo). Il learning rate verrà fissato di nuovo a 0.03 per minimizzare lo spostamento del neurone e la soglia sarà posta a 0 per permettere ad ogni neurone di partecipare alla gara.

Verranno eliminati dalla mappa radio gli access point non di proprietà della facoltà, così da evitare ingestibili modifiche di essa.

Localizzazione

Questa fase preleverà pattern di potenza, mediando 20 campioni ciascuno, in alcuni cluster dell'ambiente. Verranno rilevati molto vicino ai punti utilizzati per l'addestramento, al fine di poter riconoscere dalla mappa (fig 6.18) la loro locazione, denotando inoltre una direzione. Nella tabella 6.4 verranno riportati i risultati, evitando di fornire per semplicità l'esatto neurone vincitore, ma sottolineando il cluster ove sono stati localizzati i pattern. Saranno effettuati 4 test, con un numero sempre maggiore di access point considerati. Si sottolinea che nell'edificio ne sono presenti 7.

Come si può notare dalla tabella i risultati migliori si ottengono considerando quasi tutti gli access point. Ovviamente, maggiore è il numero degli access point e maggiore sarà l'efficienza di localizzazione. Probabilmente l'inserzione nel calcolo anche dell'access point meno potente rischia di compromettere il risultato a causa della maggiore aleatorietà di quest'ultimo, essendo probabilmente il più distante. Dovendo cercare un compromesso fra numero di access point utilizzati per la fase online ed errori dovuti all'aleatorietà di questi, grazie al test empirico si opta di utilizzare in questo ambiente i 6 access point più potenti escludendone 1 per potere utilizzare un servizio di localizzazione al massimo delle prestazioni.

Il numero di access point purtroppo non è una variabile che l'utente può scegliere e questo cabla la rete neurale. Siccome è emerso che l'access point meno potente risulta quello da scartare, si decide di cablare tale rete in modo da non considerare mai quello meno potente. Un'evoluzione dell'applicazione potrebbe essere quella di permettere all'utente di impostare questo valore.

6.3 Paragoni con altri modelli

Confronti con altri modelli, anche se previsti, non sono potuti essere realizzati per mancanza di tempo e risorse. L'efficienza della rete rimane comunque paragonabile a quella della tecnica RADAR, grazie al fatto che l'impostazione dei parametri della rete ha evitato la fusione di pattern all'interno

Tabella 6.4: Tabella di risultati dei test al secondo piano

studio:punto	4 AP	5 AP	6 AP	7 AP
4:d-S	studio4	studio4	studio4	studio4
5:b-O	studio5	studio5	studio5	studio5
5:c-S	studio6	studio6	studio6	studio6
6:f-N	studio5	studio6	studio6	studio6
6:b-N	studio9	studio6	studio6	studio6
6:e-E	studio6	studio6	studio6	studio6
6:d-N	studio6	studio6	studio6	studio6
8:d-N	studio8	studio2	studio2	studio1
8:h-O	studio8	studio8	studio2	studio1
8:a-S	studio2	studio2	studio2	studio1
8:g-S	studio2	studio2	studio2	studio1
8:b-E	studio8	studio2	studio2	studio1
9:f-O	studio9	studio9	studio9	studio9
9:c-S	studio3	studio9	studio9	studio9
9:d-S	studio9	studio9	studio9	studio9
9:e-E	studio9	studio9	studio9	studio9
10:f-N	studio7	studio7	studio10	studio10
10:c-E	studio7	studio7	studio10	studio10
10:d-S	studio10	studio10	studio10	studio10
10:e-S	studio10	studio10	studio10	studio10
11:d-E	studio11	studio11	studio11	studio11
11:g-E	studio11	studio11	studio11	studio11
11:a-N	studio11	studio11	studio11	studio11
11:b-S	studio11	studio11	studio11	studio11
corr:a-N	studio8	studio8	studio1	studio8
corr:o-N	corridoio	studio2	studio2	studio2
corr:c-O	corridoio	corridoio	corridoio	studio6
corr:v-S	studio10	studio10	corridoio	corridoio
corr:e-E	corridoio	corridoio	corridoio	corridoio
corr:q-N	corridoio	corridoio	corridoio	corridoio

corr:r-O	corridoio	corridoio	corridoio	corridoio
corr:s-S	corridoio	corridoio	corridoio	studio6
corr:v-E	studio3	corridoio	studio10	studio7
corr:i-O	studio11	studio11	studio11	studio11
corr:t-E	corridoio	corridoio	corridoio	studio2
SUCCESSI	65.7 %	68.6 %	71.4 %	62.9 %

di neuroni comuni lasciando perciò ogni neurone ad identificare un pattern d'addestramento.

In seguito alle modifiche effettuate alla funzione di attivazione dei neuroni di uscita, come già citato, si è ottenuta una gara ove ogni neurone nascosto gareggiava alla pari contro qualsiasi altro. Perciò ogni pattern di addestramento ha gareggiato solitario. Quindi si sono ottenuti gli stessi risultati che una tecnica radar, con confronto di pattern con vettori della mappa radio, poteva dare.

Seri paragoni si potrebbero fare con le evoluzioni della rete neuronale proposte, ove una taratura ottimale dei parametri e una modifica a certe caratteristiche di quest'ultima porterebbero risultati migliori rispetto RADAR. Non si dimentichi inoltre che i risultati della tecnica RADAR sono forniti per ambienti con numerosi studi, e nonostante la morfologia sia la stessa gli ambienti utilizzati per questa tesi sono divisi da pareti in cartongesso, non muratura, perciò molto meno isolanti.

Capitolo 7

Il servizio di localizzazione

7.1 Use cases

Di seguito si presentano i principali casi d'uso che l'applicazione finale andrà ad affrontare. Non avendone ancora una distribuita e fornita come servizio, attraverso questi esempi si potranno esaminare le diverse soluzioni prima di sviluppare qualcosa di concreto che utilizzi la rete neurale trattata durante i precedenti capitoli.

7.1.1 Localizzazione personale

Si immagini un utente giungere all'università e trovarsi un po' spaesato essendo il suo primo giorno di lezioni. Non è detto che questo non possa capitare anche ad un nuovo docente.

Lo studente, sapendo di dover raggiungere l'edificio, il giorno precedente aveva visitato il sito della facoltà scoprendo di un servizio nuovo rivolto alle matricole o in generale ai nuovi visitatori della struttura universitaria. Perciò, preso dal panico di perdersi all'interno dell'edificio e volendo sperimentare questa nuova soluzione software, aveva settatato la propria apparecchiatura portatile (laptop) come nel sito gli era stato suggerito. Oggi trovandosi alla porta dell'edificio sovvienne al nostro amico che la facoltà può comunicare la propria posizione e fornire una mappa di dove si trova. Egli accende il portatile e fornendo un proprio identificativo univoco riceve una mappa dell'edificio con indicata

la propria posizione. Incredibile! Ora sarà un gioco da ragazzi poter raggiungere l'aula 4 disegnata nella mappa, dove si terrà la lezione.

Questo banale esempio serve per dare un quadro delle potenzialità di un servizio tale. Infatti si pensi che l'utente non solo possa ricevere la mappa, ma anche un percorso da effettuare all'interno dell'edificio per dirigersi al punto desiderato. In realtà fuoriuscendo dall'ambito localizzativo i servizi fornibili ad uno studente potrebbe essere infinitamente vasto, si pensi solo all'accesso all'edificio controllato da un sistema di permessi, oppure ad una fornitura personalizzata di documenti o di notizie su eventi giornalieri.

7.1.2 Localizzazione di altri utenti

Se prima si era pensato ad uno studente che deve raggiungere un'aula, ora si pensi ad uno studente che deve raggiungere un professore.

Ancora una volta l'utente decide di utilizzare i servizi forniti dall'università, ma questa volta il problema risulterà quello di andare a ricevimento da un professore. Egli aveva comunicato il numero dell'aula per fare ricevimento ma una volta giunto nell'edificio, lo studente si accorge di aver perso l'appunto dove si era scritto la stanza da raggiungere. Allora connettendosi attraverso il suo laptop al server della facoltà, richiede di poter localizzare il professore che gli aveva dato appuntamento. In risposta ottiene il nome dell'aula dove il docente lo aspetta.

Anche in questo caso il servizio potrebbe essere ampiamente arricchito con mappe e percorsi di come raggiungere una persona che nel frattempo potrebbe decidere di spostarsi. Ecco allora che le informazioni dinamicamente potrebbero cambiare con il consenso di chi, di queste informazioni, ne è il proprietario (vedi sez 8.2).

7.2 Soluzione 1: Servizi costruiti su chiamate da remoto

Il problema che si viene a delineare come già detto è la scelta di come sviluppare questi servizi. Avendo sino ad ora solamente un'applicazione che fa localizzazione, servirebbe che essa non influenzi tutto un sistema di servizi, che potrebbe esulare dal fare della localizzazione.

A parte ciò, continuando ad implementare soluzioni orientate agli oggetti, per non perdere le buone caratteristiche di Java, la più semplice soluzione meno strutturata che viene alla mente è quella di un sistema a chiamate di procedure remote. Ovviamente per fare questo si dovranno utilizzare tecnologie come RMI che Java mette a disposizione, anche se la discussione su quale tecnologia utilizzare non è per ora importante volendo evidenziare il comportamento del servizio non il mezzo.

7.2.1 Localizzazione personale

Decidendo di affrontare questo problema con una chiamata a procedura remota questa sarà la discriminante per questa soluzione. Per localizzazione personale si intende ottenere una stringa o ancora meglio una mappa (un URL per essa), che indichi la posizione all'interno dell'edificio dell'utente che fa la richiesta passando al sistema remoto i dati per farlo, ovvero un pattern di potenza del luogo nel quale si trova.

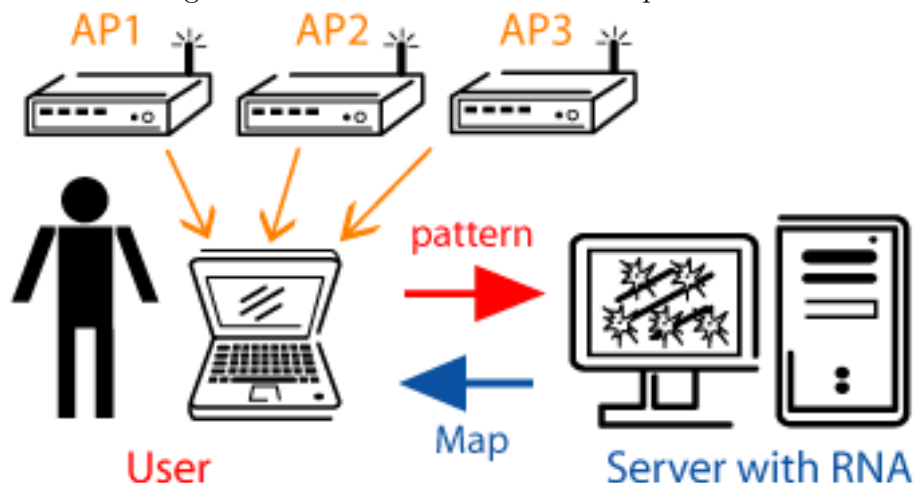
Di seguito si propongono due signature che il sistema deve mettere a disposizione per gli utenti che vogliono usufruire del servizio.

- `String getLocalCluster(Pattern pat)`
- `URL getLocalMap(Pattern pat)`

Oltre ai due metodi sopra definiti si può dare una rappresentazione grafica (fig 7.1).

Come si vede c'è interazione solo fra utente e server del servizio che può usufruire della rete neurale per ottenere il cluster dove si trova l'utente e costruire una mappa per questo che ne indichi la posizione.

Figura 7.1: Use case di localizzazione personale



7.2.2 Localizzazione di altri utenti

Questo secondo caso risulterà più complesso perchè mette in gioco più enti. Infatti l'utente non può fornire alla rete un pattern di potenze perchè in qual caso vorrebbe dire che le avrebbe rilevate dall'apparecchio dell'utente che interessa localizzare e questo non può esser possibile.

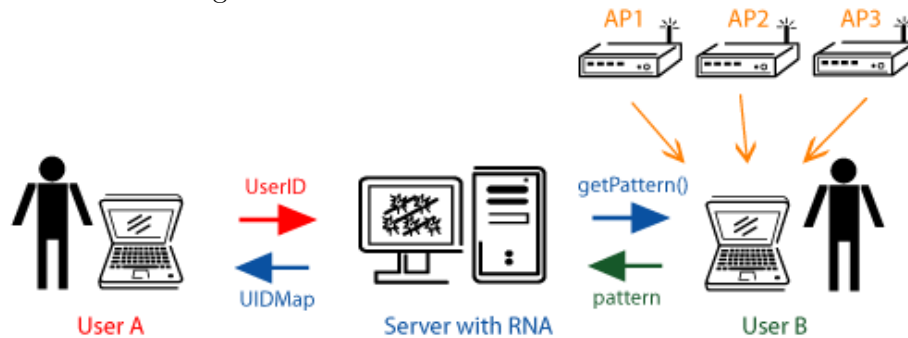
Dovrà perciò fornire una qualche stringa che permetta al server di identificare chi localizzare. Come primo ostacolo perciò ci sarà quello di definire un identificativo per ogni utente e in seguito un serie di alias che permettano a chiunque di identificare una persona. Infatti si rischia di dare ad un utente un identificativo che non corrisponderà con quello consegnato da chi fa la ricerca. Una soluzione potrebbe essere mantenere una tabella di identificativi utenti e di loro alias. Ancora una volta la richiesta potrebbe essere di una mappa o solo dell'identificativo di cluster.

- String `getLocalCluster(String UID)`
- URL `getLocalMap(String UID)`

Superato questo problema rimane il fatto che la rete deve contattare il ricercato per conoscere la sua posizione. Questa è la prima variante. Infatti la rete conoscendo l'IP della macchina del ricercato fa una richiesta ad essa per ottenere il proprio pattern di potenza. Ma allora la macchina deve

sapere che una richiesta di questo genere può avvenire in qualsiasi momento e allora si pone il problema di mantenerla in ascolto.

Figura 7.2: Use case di localizzazione altrui



Il metodo per la richiesta del pattern di un utente dovrebbe venire richiamato dal sistema di servizio che contiene la rete, per un utente con un determinato IP recuperato da una tabella degli utenti. Per ora si accantonano le problematiche che possono sorgere dal fatto che possa cambiare IP. Si potrebbe utilizzare un IP statico o il MAC della scheda wireless dell'utente. Di seguito si riporta una possibile signature.

- Pattern `getPattern()`

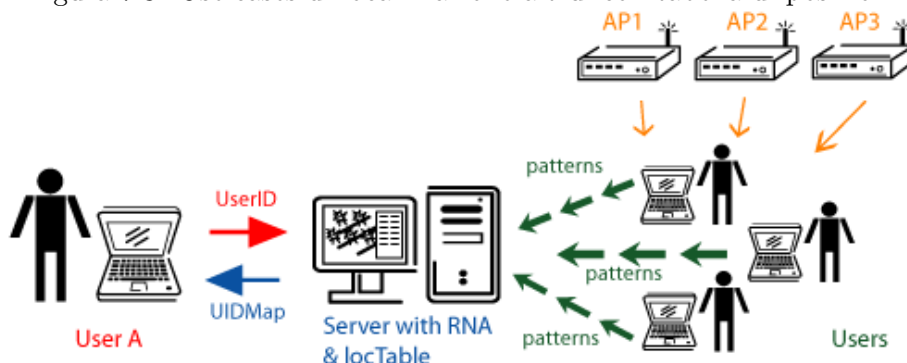
In seguito a ciò il server potrà consegnare la mappa o la locazione dell'utente ricercato a chi ne ha fatto richiesta.

La seconda variante potrebbe essere quella di mantenere in memoria una tabella di registrazione dei pattern dei vari utenti. Si pensi che ogni utente ad intervalli potrebbe aggiornare la propria posizione all'interno del server dei servizi, ad esempio in una tabella che mantiene l'accoppiata `id_utente-pattern_utente`.

In questo modo servirebbe che ogni utente costantemente contatti il server per comunicargli la propria posizione. Così facendo appena un utente richiede di localizzare un altro utente potrà essergli fornita la posizione.

Questa soluzione necessita di commenti negativi, infatti se l'aggiornamento della tabella del server da parte degli utenti, o da parte dello stesso server che potrebbe continuamente interrogarli, fosse poco frequente si rischia di fornire localizzazioni datate a chi ne richiede. D'altra parte se gli

Figura 7.3: Use cases di localizzazione altrui con tabella di posizioni



aggiornamenti della tabella fossero troppo intensi, sia che vengano eseguiti dai client sia che vengano eseguiti dal server, si rischierebbe di avere un grosso spreco temporale delle macchine impegnate all'aggiornamento. Per questo in una soluzione a chiamate a procedura remote questa scelta verrebbe abbandonata.

7.3 Soluzione 2: Servizi costruiti su TuCSon

La seconda soluzione per fornire il servizio in ambiente distribuito, verte sopra l'infrastruttura TuCSon che fornisce la possibilità di creare centri di coordinazione fra agenti ed è quello che servirebbe immaginando ogni client del servizio come un agente che rispetta il target di fornire al suo proprietario le informazioni che desidera (in questo caso luoghi di appartenenza di utenti).

Prima di capire come il sistema possa diventare ad agenti ci si preoccupa di fornire una breve introduzione all'infrastruttura TuCSon.

In seguito verrà proposta per entrambi i casi d'uso una soluzione, dove ogni utente dispone di un agente personale in grado di richiedere e usufruire di servizi. I vantaggi di questa soluzione, a discapito della precedente, consiste nel fornire al mondo delle informazioni fruibili da ogni parte dello spazio, che in un ambiente distribuito sarà inteso come una rete, con un limite massimo pari ad internet.

Inoltre TuCSon permette di non vincolare l'applicazione al buon esito delle interazioni nel distribuito. In un servizio a chiamate di procedura remota si devono già conoscere con certezza le macchine da contattare per effettuare la chiamata. Invece grazie a TuCSon, un ambiente condiviso da

tutti (come il centro di tuple) non necessita che gli agenti si conoscano fra di loro a tal punto da chiamarsi procedure l'un l'altro.

Inoltre si pensi alla caduta di un peer a cui verrebbe effettuata la chiamata a procedura. Ciò potrebbe portare in crash il sistema. Con TuCSoN si usufruisce di un sistema non bloccante che permette di non compromettere il servizio, al massimo verrebbe compromesso solamente l'agente che si trova sul peer uscito dalla rete.

L'unica complicazione da analizzare sarà la possibile caduta del nodo contenente i centri di tuple. Ciò potrebbe venire risolto con dei nodi di supporto per questo tipo di avvenimenti.

7.3.1 L'infrastruttura TuCSoN

L'infrastruttura TuCSoN proposta in alternativa a chiamate di procedure remote, potrà aiutarci nel rendere più lasco il sistema e meno vincolato rispetto al funzionamento chiamate a remoto. Infatti con TuCSoN è possibile costruire una società ad agenti in grado di interagire tra loro tramite un linguaggio logico (utilizzando tuple Prolog) in spazi dedicati alla comunicazione, vedi Centri di Tuple. Lo svincolamento citato in precedenza risiede nel fatto che la comunicazione non risente del fattore tempo, perciò un'informazione viene scritta e anche se non sono presenti contemporaneamente entrambi gli attori (agenti) di una comunicazione nulla viene bloccato. La comunicazione avrà termine quando il destinatario del messaggio leggerà l'informazione.

Per capire meglio la logica che sta dietro a questa infrastruttura si cita una fonte.

"Ovviamente l'ambiente è virtuale ed il più noto e grande ambiente virtuale è sicuramente internet; questo deve fare fronte a determinati problemi legati alla sopravvivenza ed allo svolgimento della vita di agenti singolarmente o, ancora di più, in società. Sopra l'ambiente è quindi necessario posizionare una infrastruttura che fornisca le necessarie risorse, come già discusso, e che risolva (o sia abilitante alla risoluzione) dei problemi di interazione, competizione, comunicazione, coordinazione e cooperazione. Infatti gli agenti devono essere in grado di interagire tra loro ad esempio cooperando per il raggiungimento di un goal comu-

ne, questo può facilmente implicare la necessità di coordinarsi e prima ancora di comunicare. Un meta-modello per la coordinazione necessario al fine di realizzare una infrastruttura per agenti lo possiamo trovare nello Spazio di Tuple. Questo consente ai coordinabili (agenti) di sincronizzarsi, cooperare e competere utilizzando le tuple logiche (producendone e consumandone) all'interno dello spazio di tuple. L'accesso alle tuple avviene in maniera associativa, quindi nel caso in cui un agente sia alla ricerca della tupla " $p(x)$ " non dovrà fare altro che richiedere tale tupla tramite le primitive di interazione specificandone la forma logica, ad esempio " $in(p(x))$ " consumerà una tupla siffatta nello spazio (se presente). Lo Spazio di Tuple viene considerato come media di coordinazione, le tuple logiche come linguaggio di Comunicazione ed un insieme di operazioni su tale Spazio il linguaggio di Coordinazione.

Linda è il linguaggio realizzato da David Gelernter e Nicolas Carriero che rende concreto questo modello. Una importante feature derivante dal modello è la Comunicazione generativa, ovvero le tuple generate da un agente hanno una vita, nello spazio, indipendente dall'agente che le ha generate, ed è a disposizione di tutti tramite accesso associativo. Questo implica la possibilità di avere una comunicazione disaccoppiata nello spazio e nel tempo tra agenti, [...]. Le primitive di interazione, unitamente alla loro semantica, che Linda propone sono le seguenti:

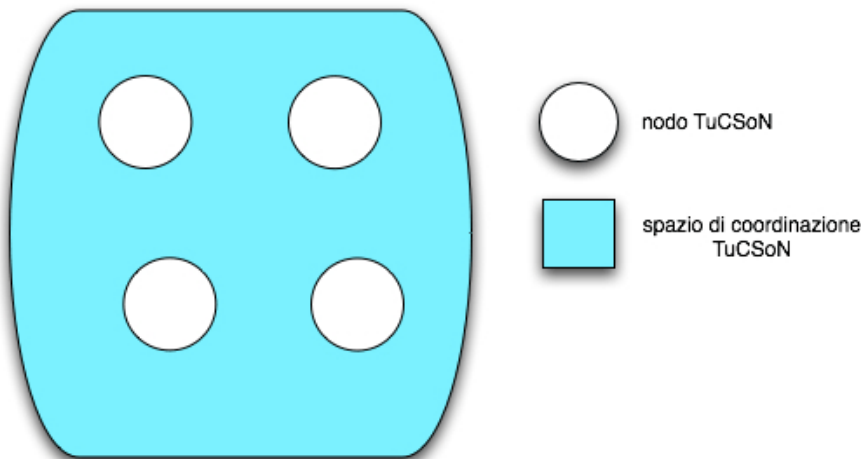
- $out(T)$ - inserisce nello spazio tuple la tupla T ;
- $in(T)$ - preleva dallo spazio tuple la tupla T ;
- $rd(T)$ - legge nello spazio tuple la tupla T ;

Estendendo il concetto di Spazio di Tuple si arriva alla definizione del Centro di Tuple con TuCSoN (acronimo di Tuple Centre Spread over the Network). Un centro di tuple, a differenza dello spazio, fornisce la possibilità di programmare il proprio comportamento, ovvero consente la definizione dinamica di artefatti di coordinazione, ovvero di regole imposte a livello di coordinazione sull'infrastruttura e che ovviamente non intaccano la

naturale indipendenza degli agenti. La coordinazione messa in gioco da TuCSoN è di tipo oggettiva ovvero, contrariamente a quella di tipo soggettiva, esterna agli agenti ed al loro comportamento; questa soluzione risulta molto interessante considerato che sgrava di compiti, differenti dal proprio goal, un agente. Il fatto che questi centri siano sparsi nella rete, all'interno di nodi, consente di ottenere artefatti di coordinazione distribuiti.

Lo spazio di coordinazione è costituito dall'insieme dei nodi TuCSoN sparsi all'interno della rete (internet). Ogni nodo consente di definire un coordination context che fornisce più centri di tuple indipendenti sui quali imporre gli artefatti di coordinazione. Per potere accedere ai centri presenti in un nodo l'agente deve ottenere il contesto; sul concetto di fornire un elemento che consente l'accesso si basa la possibilità di implementare politiche di sicurezza. [...].

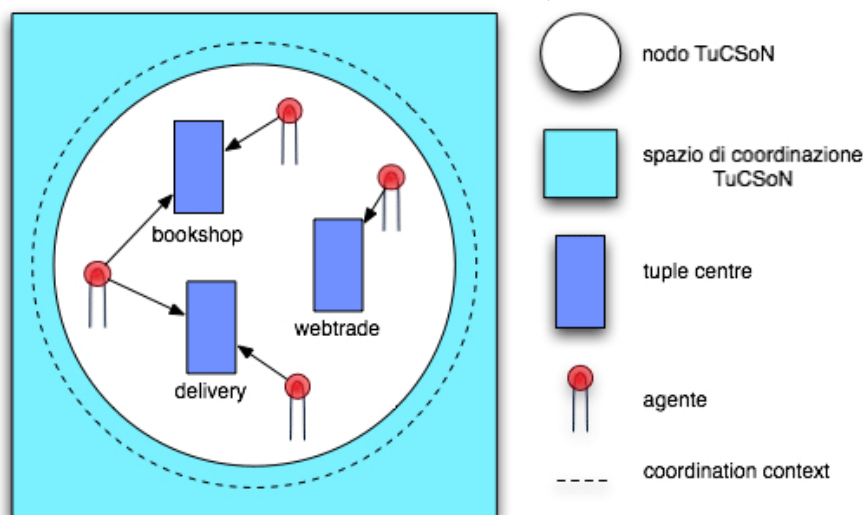
Figura 7.4: TuCSoN coordination space



Come si nota dalla figura 7.4 i nodi TuCSoN sono immersi all'interno dello spazio di coordinazione in cui gli agenti possono muoversi e migrare da nodo a nodo.

La figura 7.5 è lo zoom di un nodo in cui si notano diversi tuple centre e diversi agenti che interagiscono con almeno un centro. Il nodo è circondato dal coordination context, que-

Figura 7.5: TuCSoN node / context



sto deve essere attraversato dagli agenti esterni (mobili) al nodo per potere avere accesso ai centri di tuple. TuCSoN fornisce la possibilità di definire artefatti di coordinazione personalizzabili e general purpose; utilizzando il linguaggio di programmazione turing-equivalent ReSpecT, consente di programmare i centri, operazione eseguibile sia a set-up time che a run-time da parte di entità preposte a tale scopo e dagli agenti stessi. Quando accade una interazione il centro prima si comporta come semplice spazio di tuple, successivamente va a controllare tutte le reazioni e mette in esecuzione quelle interessate, svolgendo ogni reazione come una transazione.” [11]

Con queste premesse si passa ora a definire il sistema a servizi utilizzando gli agenti come fornitori e fruitori dei servizi, comunicando richieste e risposte attraverso il linguaggio logico.

7.3.2 Agenti del servizio

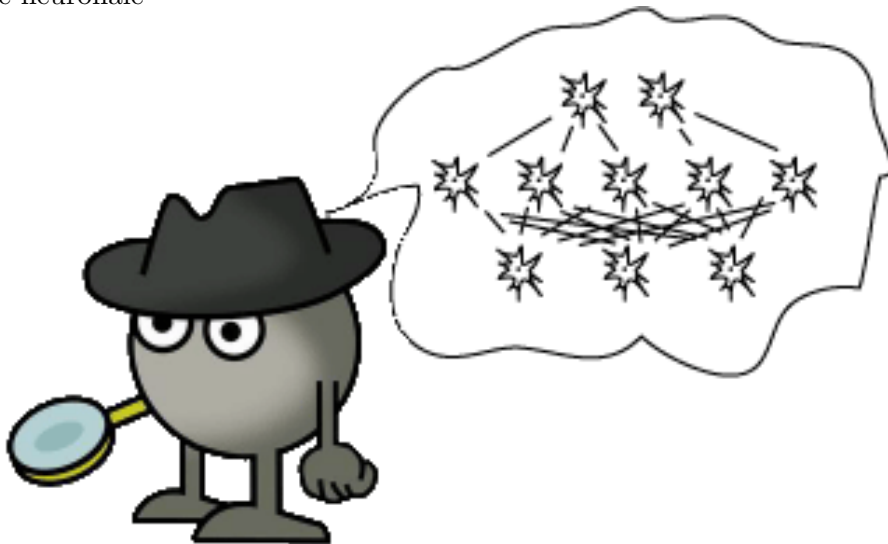
Agente Rete

C'è da stabilire per prima cosa se la stessa rete neurale è gestita da un agente continuamente presente sul server. Questa scelta è piuttosto vinco-

lata dal fatto che se qualcuno intende utilizzare la rete, che in teoria verrà ubicata su una postazione sempre on line, dalla quale essa potrà venire addestrata, dovrà poter comunicare con essa da remoto e quindi volendo utilizzare le potenzialità della seconda soluzione dovrà essere lei stessa inglobata in un agente.

Immaginando che per fornire il servizio di localizzazione si sfrutti un agente che ha in se una rete (vedi fig 7.6), questo agente dovrà stare in ascolto su un tuple center per richieste di localizzazione e quando altri agenti fanno richieste, l'agente rete deve fornire risposte.

Figura 7.6: Agente che sa fornire il servizio di localizzazione attraverso la rete neurale



Piuttosto che un tuple centre per la localizzazione, al fine di disporre un sistema più generico ed espandibile ad ogni tipologia di servizi, questo centro sarà istituito per richieste di ogni genere.

Infatti ad esso staranno in ascolto tutti quegli agenti che risolvono problemi (forniscono servizi) perciò ogni tupla catturata da essi sarà del genere

- `service_request("richiesta di servizio")`

Nel caso specifico di un agente che deve fornire localizzazione di utenti le tuple da recuperare saranno del tipo

- `service_request(whereIam('IdMittente'))`

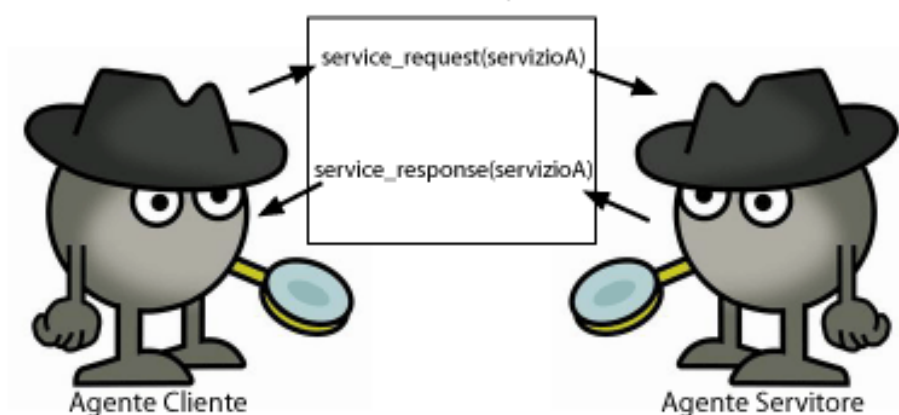
- `service_request(whereIS('IdMittente','IdRicercato'))`

e in seguito alla soluzione del problema dovrà restituire tuple del tipo

- `service_response(whereIam('IdMittente','location'))`
- `service_response(whereIS('IdMittente','IdRicercato','location'))`

Se la richiesta ha dato buoni frutti al posto di 'location' si avrà il cluster di appartenenza dell'utente. Altrimenti se non è stato trovato l'utente si avrà un 'Not found' o un 'Not recognized' se l'utente richiesto non è stato riconosciuto come uno del sistema.

Figura 7.7: Come gli agenti richiedono e forniscono servizi
Centro di Tuple



A questo punto bisognerà definire come l'agente rete risolve i due problemi che gli agenti utenti gli proporranno.

Per localizzare un utente la rete ha bisogno del suo pattern di potenza. Ci saranno due scelte intraprendibili.

La prima risulterà quella di chiedere ad ogni utente il proprio pattern di potenza quando necessita, così se l'utente vuole sapere la propria posizione comunica la potenza che lui rileva altrimenti se un utente vuole conoscere la posizione di un altro utente comunica un identificatore di modo che il fornitore di servizio possa contattare il ricercato e richiederne la potenza. Questa metodologia è performante e non necessita un grosso passaggio di informazioni, infatti esse girano solo in caso di richiesta.

Il problema però, sta nel fatto che un utente dovrà essere continuamente in ascolto, per fornire le proprie potenze se qualcuno le richiede. Questo risulta vincolante e poco appropriato perchè ogni agente utente viene a fare parte della fornitura di servizi direttamente con l'agente rete.

Volendo disaccoppiare queste due figure si arriva alla seconda soluzione. Ogni agente utente potrebbe comunicare in uno spazio comune il proprio pattern di potenze ad intervalli molto frequenti, da non mettere in crisi il sistema e allo stesso tempo da non disperdere informazioni. Siccome il sistema di prelevamento potenze non potrà superare rilevamenti ad intervalli di 100 ms e calcolando che per avere una buona approssimazione le rilevazioni hanno bisogno inoltre di una mediazione prima di essere fornite, il problema di mettere in crisi il sistema per la sovrabbondanza di informazioni in brevi intervalli di tempo non si presenta.

Le tuple che verranno prelevate per conoscere le potenze di un utente saranno del tipo

- `signal_strength('IdUtente', 'Ap1 Pow1 ; Ap2 Pow2 ; ...')`

Verrà creata perciò un pool di potenze dove l'agente rete potrà attingere per conoscere le potenze degli utenti e fornire la loro località. Questo pool potrebbe essere di interesse per altri servizi che necessitano di conoscere le stesse informazioni.

Ora però non si può dimenticare l'addestramento della rete. Infatti questa fase deve poter essere eseguita anche a rete funzionante. Perciò come addestrare la rete senza fermare il servizio che la gestisce risulta un problema da affrontare.

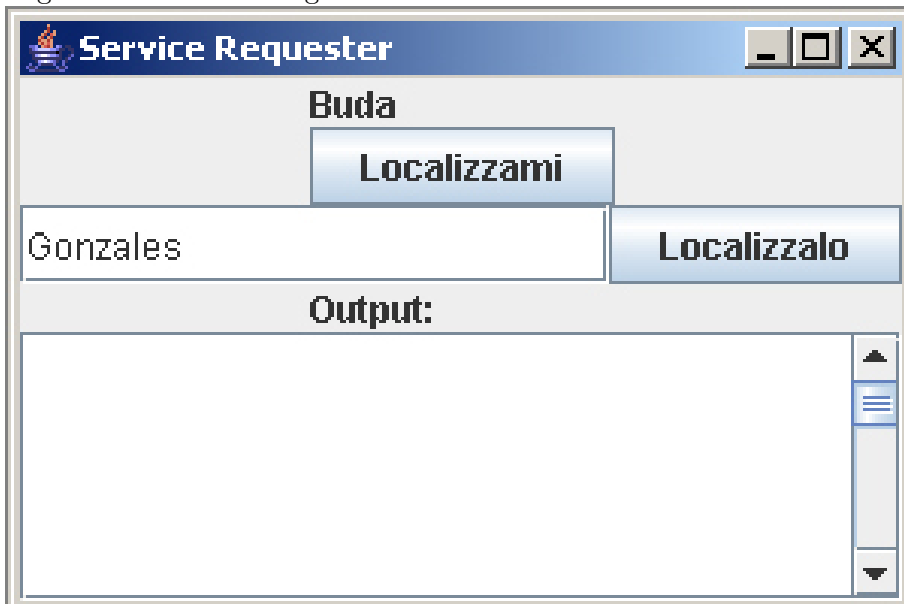
La soluzione potrebbe essere quella di addestrare la rete utilizzata dall'agente caricando la sua copia presente sotto forma di file xml e aggiornarne lo stato. Bisognerà perciò avvertire l'agente che la rete è cambiata e dinamicamente caricare la nuova rete da file ove risiede quella con i nuovi addestramenti. Ma come comunicare all'agente l'avvenuto aggiornamento della rete? O attraverso una tupla all'interno di un centro oppure attraverso un metodo dell'agente, ma questo scardina la filosofia degli agenti e del loro modo di comunicare con l'esterno.

Allora si dovrà fornire l'applicazione di costruzione e addestramento rete (vedi cap 4) di un piccolo agente che comunichi in uno spazio comune agli agenti Rete che è avvenuto un aggiornamento di stato (vedi sez 7.3.2).

Agente utente

L'agente utente potrà disporre di un'interfaccia grafica per localizzare se stesso o altri utenti. Questa interfaccia avrà grosso modo l'aspetto di fig 7.8

Figura 7.8: Interfaccia grafica di un fruitore di servizi di localizzazione



L'utente potrà sia conoscere la propria posizione sia conoscere la posizione altrui. Quindi all'avvio dell'applicazione client verrà creato un agente che potrà ascoltare le richieste dell'interfaccia attraverso un buffer che le raccoglie. Le richieste saranno come già detto di due tipi.

Le richieste verranno effettuate inserendo nel pool di servizi tuple del tipo

- `service_request(whereIam('IdMittente','location'))`
- `service_request(whereIS('IdMittente','IdRicercato','location'))`

e le risposte si otterranno raccogliendo tuple del tipo

- `service_response(whereIam('IdMittente','location'))`

- `service_response(whereIS('IdMittente','IdRicerca','location'))`

con un identificatore scelto dall'utente.

Volendo fornire all'utente maggiori informazioni sulla locazione del ricercato, si può utilizzare un sistema veloce grazie alla praticità con cui l'infrastruttura permette di associare informazioni ad altre. Infatti una semplice stringa che denota il luogo ove l'utente è stato localizzato potrebbe essere accompagnata da un'immagine che ne amplifica il significato.

L'idea più performante sarebbe quella di inserire un nuovo centro di tuple dove l'agente una volta ottenuta la locazione tramite la stringa che la denota possa accederci per avere un'immagine o meglio delle coordinate per l'immagine della mappa che l'utente già disporrebbe. La mappa potrebbero venire caricata da un agente amministrativo sul centro di tuple precedentemente citato.

Comunicandolo all'interfaccia grafica essa potrebbe disegnare un simbolo per identificare nella mappa dove si trova l'utente. Questa scelta vincola solamente ad avere a fattore comune la conoscenza della mappa, e perciò la modifica dei punti sulla mappa consisterebbe solo in una istruzione diversa del centro di tuple da parte dell'amministratore senza che l'utente necessiti di modifiche al proprio software.

In questa tesi per ovvi motivi di tempo si opta per una soluzione più semplice. Preimpostando una serie di immagini che indicano i punti inerenti ai cluster sulla mappa, è l'utente che conoscendo la clusterizzazione che effettuerà la rete, in seguito alla stringa di localizzazione visualizza l'immagine corretta. Questa scelta risulta molto vincolante ma permette di ottimizzare prestazione e tempi.

Questo sarà il compito principale dell'agente, senza però dimenticare che l'agente dovrà costantemente riempire un centro di tuple con l'informazione della potenza propria rilevata attraverso la tupla

- `signal_strength('IdUtente', 'Ap1 Pow1 ; Ap2 Pow2 ; ...')`

Ad ogni nuova rilevazione, la tupla precedente, riguardante l'utente, verrà sostituita con una nuova. Questa potrebbe essere anche fonte d'informazione sulla presenza dell'utente nel sistema. Infatti il centro di tuple

potrà eliminare la tupla inerente ad un ente se si arresta l'aggiornamento dei valori.

L'agente utente sarà indipendente dall'interfaccia grafica. Infatti un altro sistema a richieste di servizio, questa volta locale, rappresentato da un semplice buffer, permetterà all'interfaccia di inserire richieste di servizio che l'agente andrà a depositare nel centro di tuple predisposto. Questa scelta è stata percorsa per non compromettere il funzionamento dell'interfaccia per richieste che non dassero risposta istantanea. L'agente risponderà attraverso un messaggio o una mappa sull'interfaccia grafica ma raccoglierà le richieste dal buffer. Se il buffer si riempisse a causa della lentezza dell'agente le successive richieste andranno perdute ma l'interfaccia continuerà a funzionare avendo un suo thread autonomo.

Agente amministratore

Quest'ultimo agente sarà inserito nell'applicazione di addestramento della rete per comunicare all'agente rete l'avvenuta modifica della struttura neuronale in seguito all'inserimento di nuovi pattern di potenza. Perciò quest'agente inserirà in un centro di tuple dedicato una tupla del tipo

- `updateNet('IDrete','URL_file_specifico')`

Possiamo avere a disposizione molti agenti rete identici, per evitare che il crollo di uno mandi in crisi il sistema. In questo caso si necessita una politica di lettura di questa informazione per non consumare la tupla e rischiando di non renderla più disponibile agli altri.

Ma dove precisamente inserire l'agente? L'applicazione è costituita dalla console, dal modulo xml, dalla rete e dalle varie viste, tutte gestite da un gestore che le utilizza. Sarà perciò a questo punto il gestore il candidato più plausibile ove iniettare l'agente e permettergli di inserire nel centro di tuple l'informazione sull'aggiornamento della rete.

7.3.3 Centri di tuple identificati

In conclusione i centri di tupla identificati risultano quattro. Nelle sezioni che seguono si riportano le principali caratteristiche, gli agenti che ci interagiranno e le tuple che conterranno.

Centro 'services'

Questo centro di tupla, sarà quello più generico, per permettere ad altri servizi, non solo quello per la localizzazione, di essere fruibili agli utenti con agenti. In questo pool verranno inserite richieste di servizio e risposte alle richieste. I server e i client dei servizi, in codesto caso quello di localizzazione personale e quello di localizzazione altrui, saranno rispettivamente gli agenti rete e gli agenti utenti (agenti personali rappresentanti una persona).

Le tuple che essi inseriranno e recupereranno dal centro saranno le seguenti, come già visto:

- `service_request(whereIam('IdMittente'))`
- `service_request(whereIS('IdMittente','IdRicerca'))`
- `service_response(whereIam('IdMittente','location'))`
- `service_response(whereIS('IdMittente','IdRicerca','location'))`

Centro 'signal_strength'

Il seguente centro di tuple sarà quello contente le potenze di ogni utente presente nella facoltà virtuale creata da questo sistema. Come già sottolineato ogni utente inserirà il proprio pattern di potenze aggiornato per permettere la propria localizzazione.

Ad ogni inserimento, la vecchia tupla verrà sostituita con una nuova. Un regola del centro di tuple potrebbe infine indicare quando un utente esce dal sistema ovvero quando l'attesa di aggiornamento supererà un limite stabilito.

Gli agenti che utilizzano questo centro saranno dunque sia tutti gli utenti che lo aggiornano, sia l'agente rete che legge i valori di potenza per fare localizzazione. Quest'ultimo infatti non preleverà le tuple ma le leggerà soltanto.

Le tuple presente nel centro saranno dunque del tipo seguente

- `signal_strength('IdUtente', 'Ap1 Pow1 ; Ap2 Pow2 ; ...')`

Sarà la rete ad inserire nei pattern valori di potenza nulla per quegli access point che l'utente non rileva ma indispensabili alla rete per creare un corretto pattern di localizzazione.

Centro 'update'

In questo pool di tuple risiedono le informazioni sul file che contiene la rete con l'addestramento più recente.

Sarà l'agente inserito nell'applicazione di addestramento rete a aggiornare questo pool inserendo una tupla nuova che fa riferimento al file corretto. L'agente rete utilizzerà questo centro solamente per leggere il nome del file e creare una nuova rete aggiornata.

- `updateNet('IdRete','URL_file_specifico')`

Centro 'userIds'

Quest'ultimo centro interesserà l'agente rete per conoscere su quale utente è stata fatta la richiesta. Infatti un utente può essere riconosciuto sotto diversi alias mentre le potenze fornite nel pool inerente appartengono ad un solo identificativo. Perciò attraverso questo centro si potranno fornire più alias per ogni identificativo.

Questo potrà essere fatto sia dagli agenti utenti, sia da un amministratore che 'manualmente' inserisce tuple al centro. Chi leggerà da questo sarà solamente l'agente rete al fine di conoscere la vera identità di un nominativo.

Di seguito si riportano le tuple che caratterizzano questi alias

- `alias('IdUtente','AliasUtente')`

Capitolo 8

Conclusioni

8.1 Considerazioni finali

Obiettivo di questa tesi è stato quello di costruire un sistema di localizzazione indoor attraverso l'utilizzo di tecniche di SceneAnalysis inserite nel comportamento di una rete neurale evolutiva. L'output finale sarà composto da un sistema a servizi per permettere di utilizzare la rete neurale in un ambiente distribuito. Il percorso effettuato per ottenere l'applicazione finale ha permesso di venire a conoscenza delle problematiche tipiche di un ambiente indoor e di proporre nuove soluzioni da applicare direttamente al sistema o altrimenti ad evoluzioni future.

Dopo aver testato il funzionamento dei servizi che il sistema mette a disposizione, e constatati i risultati già ottenuti nelle prove precedenti (sez 6.2.4) si passa a tirare le somme di questo progetto.

Sicuramente il tempo non ha permesso di considerare ogni possibile strada da intaprendere per perfezionare il tool, che non è stato pienamente utilizzato in tutte le sue caratteristiche. Infatti la rete è stata testata solo per pochi parametri, ben più mirata sarebbe stata la scelta di questi avendo la possibilità di eseguire un numero maggiore di test.

Inoltre la rete neurale è stata utilizzata solo in parte, ovvero nei test eseguiti si è evitato di permettere l'unificazione di pattern all'interno di neuroni nascosti, questo per analizzare il tool in maniera più semplice possibile. Un'evoluzione futura potrebbe essere proprio l'analisi di questa caratteristica volontariamente non testata.

Non si dimentichi che la rete neuronale è stata costruita in modo da fornire gli strumenti necessari per sviluppare qualsiasi evoluzione specifica mantenendo le caratteristiche a fattore comune individuate per ogni generica rete potendo perciò riutilizzare parte del progetto.

Il funzionamento della rete potrebbe venire perfezionato in seguito alle idee scaturite durante lo studio dei risultati, si veda per questo la sez 6.2.3, ove sono state proposte diverse varianti per ottenere migliori risultati dall'applicazione. Non si dimentichi che queste politiche sono strettamente legate alla morfologia dell'edificio in cui si utilizza il servizio.

Il servizio costruito potrebbe divenire la base di un sistema più ampio applicabile ad altre situazione anche non strettamente legate al tema della localizzazione. Infatti come già per la rete, i servizi sono stati concepiti con uno sguardo al futuro, ovvero con la capacità di implementare nuove soluzioni con le minime modifiche al progetto già sviluppato.

8.2 Privacy

Per finire un breve paragrafetto aiuterà a capire come questo progetto non debba uscire dai termini della legalità.

Ovviamente un sistema così innovativo come quello utilizzato per fare localizzazione in questa tesi non è stato ancora preso in considerazione dal garante della privacy nazionale, organo istituzionale per garantire la riservatezza di informazioni sensibili riguardanti i cittadini.

Ci si avvarrà di comunicati stampa
(<http://www.garanteprivacy.it/garante/doc.jsp?ID=1109493>)
per controllare il corretto utilizzo degli RFID, microchip intelligenti utilizzati anche per localizzazione di persone. Le stesse regole, per analogia, verranno probabilmente applicate se il sistema che utilizza WLAN estenderà la sua diffusione.

Si riporta in calce parti del documento stilato dal Garante.

"Etichette intelligenti": il Garante individua le garanzie per il loro uso Quando si trattano dati personali, i cittadini devono essere informati, esprimere un libero consenso e poter disattivare i chip.[..]

Alcuni impieghi di questa tecnologia - che non si limitino a tracciare il prodotto per garantire l'efficienza del processo di produzione industriale - possono costituire una violazione del diritto alla protezione dei dati personali e determinare forme di controllo sulle persone: con l'uso di Rfid si potrebbero, infatti, raccogliere innumerevoli dati sulle abitudini dei consumatori a fini di profilazione o essere in grado di tracciare i percorsi effettuati dagli stessi, controllarne la posizione geografica o verificare quali prodotti usa, indossa, trasporta.

Ovviamente non si può parlare nel sistema che utilizza WLAN di informazioni inerenti agli acquisti o ai prodotti che indossano un microchip, ma si può dedurre per analogia la preoccupazione di conoscere le abitudini di persone, rivelando gli ambienti dove risiedono.

Informativa. Le persone devono essere adeguatamente informate dell'utilizzo di sistemi Rfid, così come dell'esistenza dei lettori ottici che attivano l'etichetta. La presenza di avvisi nei luoghi nei quali le tecniche Rfid sono utilizzate non esime da apporre informativa sugli stessi oggetti e prodotti che recano le etichette intelligenti.

Consenso. Un soggetto privato che utilizza Rfid trattando dati personali può farlo solo con il consenso espresso e specifico degli interessati, a meno che ricorra in casi particolari uno degli altri presupposti di legge. Il consenso non è valido se ottenuto con pressioni o condizionamenti sull'interessato.[..]

Accesso a determinati luoghi o a posti di lavoro. Nei casi di impiego di Rfid per la verifica di accessi a determinati luoghi riservati devono essere predisposte idonee cautele per i diritti e le libertà delle persone. In particolare: per i luoghi di lavoro va rispettato quanto previsto dallo Statuto dei lavoratori che vieta l'utilizzo di impianti per il controllo a distanza dei lavoratori; per l'accesso occasionale di terzi a determinati luoghi occorre predisporre un meccanismo che, nel caso di indisponibilità ad usare

Rfid da parte dell'interessato, gli permetta comunque di entrare nel luogo in questione.[..]

Misure di sicurezza. Chi utilizza etichette intelligenti e tratta dati personali ha l'obbligo di adottare misure di sicurezza per ridurre i rischi di distruzione, perdita, accesso non autorizzato o manomissione dei dati conservati.

Un secondo documento, in seguito ad uno studio effettuato dal garante, definisce ancora meglio le regole da adottare. Si pensi sempre all'analogia di queste regole applicabili ad un sistema di localizzazione WLAN.

[..] Infine, i rischi per la vita privata dei cittadini possono accrescersi nel caso di un'integrazione della RFID con infrastrutture di rete (telefonia, Internet, ecc.).

È quindi necessario che l'implementazione e l'utilizzo della RFID, ove si configuri un trattamento di dati personali, avvenga nel rispetto dei principi dettati dal Codice e, in particolare, delle libertà, dei diritti fondamentali e della dignità degli interessati (art. 2, comma 1, del Codice).[..]

Tutto ciò premesso, e nella consapevolezza della prevedibile diffusione della RFID in diversi settori ed impieghi, possono sin da ora distinguersi -con riferimento al profilo del consenso- alcuni casi specifici:

[..]

d) nei casi di impiego di RFID per la verifica di accessi a determinati luoghi riservati devono essere predisposte idonee cautele per i diritti e le libertà degli interessati. In particolare,

d1) ove si intenda utilizzare tali tecniche per verificare accessi a luoghi di lavoro, o comunque sul luogo di lavoro, va tenuto

conto che lo Statuto dei lavoratori vieta l'uso di impianti e apparecchiature per finalità di controllo a distanza dell'attività dei lavoratori e, nel caso in cui il loro impiego risulti necessario per altre finalità, prescrive alcune garanzie (art. 4 l. 20 maggio 1970, n. 300; art. 114 del Codice) alle quali si affianca l'osservanza dei richiamati principi di necessità, finalità e proporzionalità del trattamento dei dati;

d2) qualora la RFID sia adoperata per controllare l'accesso occasionale di terzi a determinati luoghi, occorre predisporre un meccanismo che, in caso di indisponibilità dell'interessato, permetta a quest'ultimo di entrare comunque nel luogo in questione, con eventuale adozione -e solo se necessario- di misure di cautela rimesse alla ragionevole valutazione del titolare.

E' chiara quindi l'esigenza di informare ogni utente del rischio che incorre adottando il sistema nell'esporre le propri informazioni personali. Un consenso deve necessariamente essere richiesto prima di attuare localizzazione su di una persona.

L'applicazione potrebbe in futuro avvalersi di un filtro per permettere ad utenti di essere visti soltanto da determinati altri, si pensi ad una visibilità ristretta fra professori.

8.3 Ringraziamenti

Ringrazio per questa tesi gli Ingegneri che l'hanno sostenuta a partire dal caro relatore Pablo Gonzales, il mio più stretto collaboratore Matteo Lucchi, i telecomunicazionisti che spesso mi hanno tolto da parecchie grane del loro settore Gianluigi Liva, Enrico Paolini, Davide Dardari, gli informatici che sempre mi hanno sostenuto per dettagli tecnici e non, Alessandro Ricci e Andrea Omicini nonchè il caro amico Nicola Zaghini. Si ricorda inoltre Claudia Cevenini per le dritte sulla legalità del sistema.

Ringrazio inoltre i miei genitori, la mia ragazza, parenti e amici tutti che con il loro sostegno mi hanno supportato e sopportato in questo lungo cammino.

Sperando di non aver dimenticato nessuno confido in uno sviluppo ulteriore dell'argomento affinché la tecnologia e la scienza non siano mai arrestate nel loro cammino al servizio dell'umanità.

Bibliografia

- [1] Bazzoli Francesco. Caratterizzazione statistica di ambienti indoor per sistemi wireless di posizionamento. Ingegneria delle telecomunicazioni, Università degli studi di Bologna, 2005.
- [2] Gramellini Maria Chiara. Reti neurali applicate ai sistemi di localizzazione radio. Ingegneria dei sistemi e delle tecnologie dell'informazione, Università degli studi di Bologna, 2005.
- [3] Gramellini C., Gonzalez P., and Dardari D. Una rete neurale per la localizzazione a rf in sistemi wlan. Technical report, Università degli Studi di Bologna, 2005.
- [4] Zamagni Guido. Realizzazione di moduli software per la localizzazione indoor mediante mappe di potenza. Ingegneria delle telecomunicazioni, Università degli studi di Bologna, 2005.
- [5] Kishan Mehrotra, Chilukuri K. Mohan, and Sanjay Ranka. *Elements of Artificial Neural Networks (Complex Adaptive Systems)*. MIT Press, 1996.
- [6] Teuvo Kohonen. *Self-Organization and Associative Memory*, volume 8 of *Springer Series in Information Sciences*. Springer Verlag, 3rd edition, 1989.
- [7] Teuvo Kohonen. *Self-Organizing Maps*. Springer Verlag, 3rd edition, 2000.
- [8] Sánchez R., González P.P., and Riquenes A. A neural network to learn patterns from non-stationary environments. Playa del Carmen, Q.R.,

-
- México, 1995. Proceedings The Firts Joint Mexico-US International Workshop on Neural Networks and Neurocontrol. pagine 310-325.
- [9] Marquez Maria del Carmen. Analisis de patrones de expresion de genes con redes neuronales artificiales. Master's thesis, Universidad Nacional Autonoma de Mexico, 2005.
- [10] Cinarelli Davide. Algoritmi e misure sperimentali per la localizzazione mediante sistemi wireless 802.11. Ingegneria delle telecomunicazioni, Università degli studi di Bologna, 2005.
- [11] Zaghini Nicola. Un linguaggio dichiarativo per l'ingegneria di sistemi ad agenti nell'impresa virtuale. Ingegneria dei sistemi e delle tecnologie dell'informazione, Università degli studi di Bologna, 2005.

Elenco delle figure

1.1	Architettura classica di una rete WLAN	11
3.1	Architettura del neurone artificiale	24
3.2	Architettura tipica di una rete winner-take-all	28
3.3	Architettura tipica di una rete winner-take-all	29
3.4	Architettura tipica di una rete dinamicamente evolutiva . . .	35
4.1	Struttura macrocomponenti del sistema	41
4.2	Struttura rete Evolutiva	44
4.3	Struttura proposta per l'applicazione	46
4.4	Comportamento dell'applicazione	47
4.5	Tabella delle interazioni	48
4.6	Iniezione delle funzioni	52
4.7	UML dell'Interfaccia della Rete	52
4.8	UML della classe della Rete Astratta	53
4.9	Algoritmo d'addestramento della rete Supervisionata	55
4.10	Rimpicciolimento del raggio di un neurone	56
4.11	Unificazione di un pattern col neurone	57
4.12	UML della classe che rappresenta la Rete Evolutiva Supervi- sionata	59
4.13	Diagramma di sequenza del funzionamento online della rete .	61
4.14	Funzioni dello strato d'ingresso	62
4.15	Funzioni dello strato nascosto	62
4.16	Funzioni dello strato d'uscita	63
4.17	UML della classe Neurone	65
4.18	UML Pattern	66
4.19	UML Pattern con Cluster d'appartenenza	67

4.20	UML della classe che rappresenta la Rete aperta	68
4.21	Modello matriciale dei dati di una rete	69
4.22	Console grafica per l'amministratore della rete	70
4.23	Esempio file XML per salvataggio rete	75
4.24	Struttura finale del sistema	77
4.25	Architettura finale del sistema	78
4.26	Gui dell'applicazione rete neuronale	79
5.1	Architettura dell'applicazione	83
5.2	Dall'applicazione allo strato fisico	85
5.3	Bozza d'interfaccia prelevamento potenze	86
5.4	Struttura rilevazioni	88
5.5	Bozza d'interfaccia prelevamento potenze	89
5.6	Esempio di come vengono salvate le potenze su un file XML	90
6.1	Pianta secondo piano dell'edificio in via Venezia	94
6.2	Pianta dell'edificio al piano terra, ala nord	97
6.3	Primo test di localizzazione al piano terra	99
6.4	Secondo test di localizzazione al piano terra	100
6.5	Terzo test di localizzazione al piano terra	100
6.6	Quarto test di localizzazione al piano terra	101
6.7	Quinto test di localizzazione al piano terra	101
6.8	Pianta dell'edificio al primo piano, ala sud	103
6.9	Primo test di localizzazione al primo piano	106
6.10	Secondo test di localizzazione al primo piano	107
6.11	Terzo test di localizzazione al primo piano	108
6.12	Quarto test di localizzazione al primo piano	109
6.13	Quinto test di localizzazione al primo piano	110
6.14	Sesto test di localizzazione al primo piano	111
6.15	Settimo test di localizzazione al primo piano	112
6.16	Ottavo test di localizzazione al primo piano	113
6.17	Punti di prelievo potenze nel test per le direzioni	119
6.18	Punti di prelievo di potenze al secondo piano, ala sud	122
7.1	Use case di localizzazione personale	130
7.2	Use case di localizzazione altrui	131

7.3	Use cases di localizzazione altrui con tabella di posizioni . . .	132
7.4	TuCSon coordination space	135
7.5	TuCSon node / context	136
7.6	Agente che sa fornire il servizio di localizzazione attraverso la rete neurale	137
7.7	Come gli agenti richiedono e forniscono servizi	138
7.8	Interfaccia grafica di un fruitore di servizi di localizzazione . .	140