Università degli Studi di Padova

Facoltà di Ingegneria

Corso di Laurea Specialistica in
Ingegneria delle Telecomunicazioni

Tesi di Laurea

APPLICAZIONE DELL'ALGORITMO DI NESTEROV ALLA TECNICA DI COMPRESSIVE SENSING IN RETI DI SENSORI RADIO

Relatore: Ch.mo Prof. Michele Rossi

Correlatori: Ing. Giorgio Quer, Ing. Marco Visonà

Laureando: Davide Zordan

Padova, 20 Aprile 2010

Anno Accademico 2009-2010

UNIVERSITÀ DEGLI
STUDI DI PADOVA

FACOLTÀ DI
INGEGNERIA

CORSO DI LAUREA SPECIALISTICA IN
INGEGNERIA DELLE TELECOMUNICAZIONI

TESI DI LAUREA

APPLICAZIONE DELL'ALGORITMO DI NESTEROV ALLA TECNICA DI COMPRESSIVE SENSING IN RETI DI SENSORI RADIO

RELATORE: *Ch.mo Prof. Michele Rossi*

CORRELATORI: *Ing. Giorgio Quer, Ing. Marco Visonà*

LAUREANDO: *Davide Zordan*

Padova, 20 Aprile 2010

Anno Accademico 2009-2010

CORSO DI LAUREA IN
INGEGNERIA DELLE
TELECOMUNICAZIONI

DEPARTMENT OF
INFORMATION
ENGINEERING
UNIVERSITY OF PADOVA

DIPARTIMENTO DI
INGEGNERIA
DELL'INFORMAZIONE

Sommario

In questa tesi si analizza una tecnica di compressione e ricostruzione di segnali, il Compressive Sensing (CS), e la sua applicazione in un protocollo innovativo per la raccolta e la ricostruzione dei dati di una rete di sensori radio. Il CS si basa su due assunzioni: la possibilità di rendere sparso il segnale attraverso una qualche trasformazione, e l'alto grado di incoerenza tra questa trasformazione e il metodo con cui i dati vengono raccolti. Nella procedura per il recupero dei dati si deve far fronte ad un problema di ottimizzazione convessa, risolvibile mediante algoritmi di ricerca del minimo della funzione norma ℓ_1. In questa tesi si è studiato e implementato l'algoritmo di Nesterov, un algoritmo robusto, che si adatta al problema del Compressive Sensing e che garantisce prestazioni elevate.
La tecnica del CS è stata implementata e inclusa in un framework per il controllo e la gestione di reti di sensori, di cui sono state verificate le prestazioni tramite simulazioni. I risultati ottenuti sono stati inoltre verificati tramite alcuni test simulativi con segnali reali, collezionati da diversi testbed di reti di sensori.

Indice

Elenco delle figure

Introduzione

Una rete di sensori radio, o wireless sensor network (WSN) è una rete costituita da piccoli dispositivi autonomi, distribuiti nello spazio e in grado di cooperare al monitoraggio di condizioni fisiche o ambientali, come temperatura, suono, vibrazione, pressione, movimento o presenza di sostanze inquinanti [1]. Lo sviluppo delle reti di sensori radio è stato spinto da applicazioni militari come il controllo di aree di guerra o l'identificazione di sottomarini [2]. Al giorno d'oggi sono utilizzate in molti campi di applicazione industriale e civile, compreso il controllo di processi industriale, il controllo territoriale e delle condizioni ambientali, applicazioni sanitarie, home automation e il controllo del traffico.

Ci sono però una serie di differenze molto importanti tra una WSN e una qualsiasi altra tipologia di rete, per cui non è possibile utilizzare le tecniche standard per le comunicazioni wireless. Le principali caratteristiche che rendono una WSN diversa dalle altre tipologie di rete sono:

- Il numero di nodi che compongono una rete di sensori può essere di alcuni ordini di grandezza più grande del numero di nodi in una rete ad hoc;
- I nodi sensore sono disposti con un'alta densità;
- I nodi sensore sono soggetti a guasti;
- La topologia di una rete di sensori cambia frequentemente;
- I nodi sensore hanno una capacità di calcolo e di memoria limitata, unitamente ad una fonte di energia limitata.

Mentre molte ricerche sono state effettuate al fine di sviluppare protocolli di comunicazione robusti e in grado di sopperire alle mancanze in termini di affidabilità delle reti di sensori, un campo che necessita di ulteriori investimenti è quello dell'efficienza e del risparmio energetico.
In alcuni scenari infatti, la sostituzione delle batterie dei sensori potrebbe essere impossibile, ad esempio per nodi posizionati sul fondo dell'oceano [3], o comunque molto costosa, dato che in ogni caso è richiesto l'intervento di un operatore. La vita di un sensore è dunque strettamente legata alla durata della batteria.

In una rete di sensori multihop inoltre, ciascun nodo svolge il duplice ruolo di generare dati e di instradare i dati di altri sensori verso il punto di raccolta. Il disfunzionamento di pochi sensori quindi può provocare cambiamenti significativi nella topologia di rete e potrebbe richiedere una riorganizzazione del routing dei pacchetti e della rete stessa.
Dato che il maggior consumo di energia da parte di un nodo sensore è dovuto alla comunicazione, è fondamentale l'utilizzo di tecniche di aggregazione e compressione dei dati che permettano il recupero di un segnale da una quantità minima di letture. Ad ogni round di raccolta dati parte dei sensori rimangono inattivi, consentendo un significativo risparmio delle risorse energetiche. I restanti sensori trasmettono il loro dato al punto di raccolta (*sink*), dove si introduce una procedura di recupero dei dati. Questa procedura deve essere in grado di recuperare il segnale completo minimizzando l'errore di ricostruzione. Ai sensorio viene inviata una probabilità di trasmissione, secondo la quale devono decidere se trasmettere o meno la propria lettura ad ogni istante temporale. Tutta la complessità della procedura risiede nel punto di raccolta, che però non risente delle stesse problematiche dei sensori dato che solitamente esso dispone sia di una alimentazione esterna che di una buona capacità di calcolo.

Una tecnica matematica che permette l'acquisizione e la ricostruzione di un segnale utilizzando una quantità minima di informazione, minore di quanto imposto dal teorema del campionamento, è Compressive Sensing (CS)[4], tecnica che verrà analizzata e implementata in questa tesi. Questa tecnica si basa sulla possibilità di rappresentare il segnale di interesse attraverso un segnale sparso in un diverso dominio. CS permette di sfruttare la correlazione spaziale e temporale dei segnali fisici che solitamente sono oggetto di misurazioni nelle reti di sensori (quali temperatura, umidità o altre grandezze), per riuscire a ridurre il numero minimo di campioni necessario a ricostruire l'intero segnale.

Per la ricostruzione del segnale completo, CS richiede la ricerca della soluzione più sparsa che soddisfi un sistema di vincoli costruiti sulla base alle tecniche di routing utilizzate e di una matrice di trasformazione. Sia la modalità con cui i dati vengono raccolti che il metodo con cui il segnale viene reso sparso infatti sono descritti da matrici, e sulla base di queste matrici viene costruito un sistema di vincoli che i dati ricostruiti devono rispettare [5]. Per la ricerca della soluzione più sparsa si può far uso di diversi algoritmi di programmazione convessa [6, 7, 8, 9]. In questa tesi è riportato un particolare algoritmo che risulta avere ottime prestazioni, l'algoritmo di Nesterov. In particolare questo metodo è stato studiato, implementato e testato in ambiente simulativo.

L'algoritmo di Nesterov, applicato alla tecnica CS è stato poi integrato in un framework per la gestione e il controllo di reti di sensori sviluppato nella tesi di laurea dall'ing. Marco Visonà [10], correlatore per questo lavoro. Un modulo che permetta di attuare il Compressive Sensing è stato scritto e integrato nel framework *WSNControl* e sono stati effettuati alcuni test su segnali reali raccolti da varie WSNs.

Il resto della tesi è organizzato come segue:

Capitolo 1: introduzione al framework WSNControl e descrizione delle sue funzionalità prima e dopo l'inserimento del modulo che effettua il Compressive

Sensing;

Capitolo 2: descrizione della tecnica CS unitamente alla Principal Component Analysis;

Capitolo 3: presentatazione del metodo di Nesterov e dell'algoritmo NESTA dopo i necessari richiami matematici;

Capitolo 4: valutazione delle prestazioni dell'algoritmo proposto e confronto con quelle di altri metodi presenti in letteratura attraverso simulazioni in ambiente di calcolo numerico;

Capitolo 5: presentazione del modulo *Rebuilder*, dei metodi di raccolta e ricostruzione dati e dei metodi di stima dell'errore commesso;

Capitolo 6: descrizione degli scenari in cui si sono effettuati i test e presentazione dei risultati ottenuti;

Capitolo 7: conclusioni e sviluppi futuri.

1 WSNControl

Il software di gestione e monitoraggio *WSNControl* rientra all'interno del progetto europeo SENSEI [11]. Tale progetto, la cui rappresentazione concettuale è riportata in in Figura 1.1, mira a realizzare un framework di proporzioni globali attraverso il quale sia possibile accedere ai servizi offerti da reti di sensori e attuatori, dette anche *Wireless Sensor & Actuator Networks* (WSAN).
Le applicazioni di questa tecnologia sono molteplici e sono tutte nell'ottica di fornire intelligenza all'ambiente che ci circonda. Componente essenziale del pro-

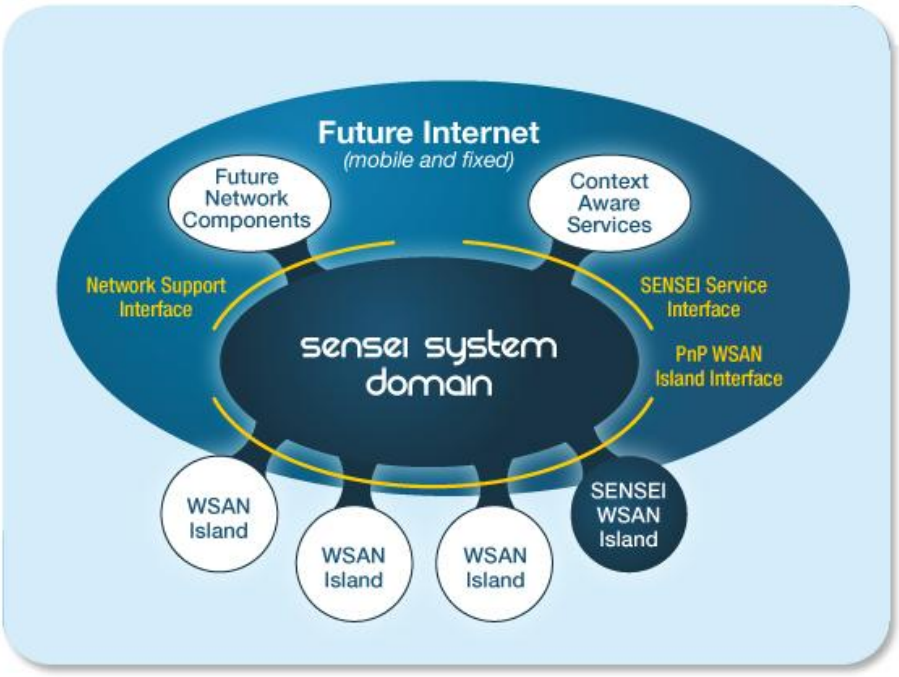


Figura 1.1: Obiettivo del progetto SENSEI

getto SENSEI è un sistema di acquisizione dati e monitoraggio di reti di sensori che sia affidabile e che abbia buone prestazioni.

Un sistema siffatto può essere realizzato secondo il modello riportato in Figura 1.2, che mostra l'architettura generale di un sistema di monitoraggio.

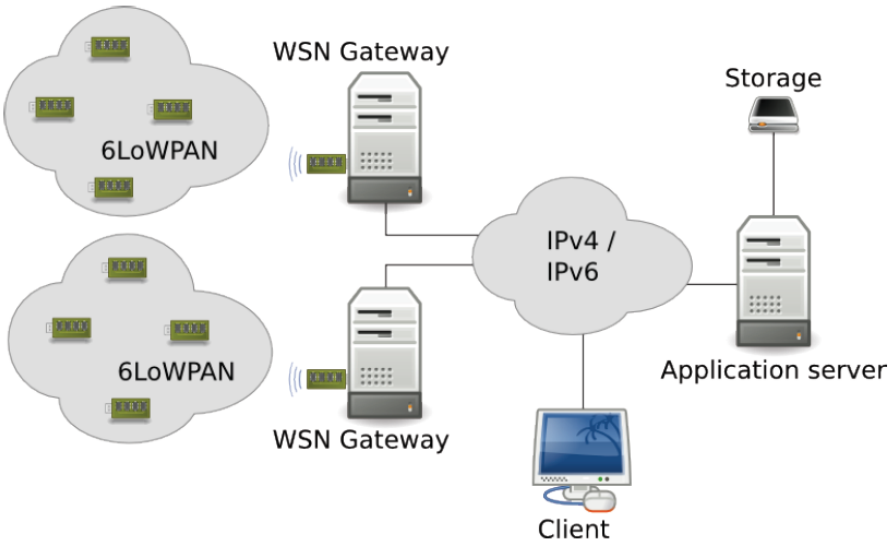


Figura 1.2: Architettura generale di un sistema di monitoraggio.

Questa architettura consente di collegare una o più reti di sensori ad un punto di raccolta dati, detto *WSN Gateway*, in comunicazione con un *Application Server*. Gateway e application server sono collegati alla rete Internet basata sul protocollo IPv4 o, nel futuro, alla rete basata su IPv6.

Sebbene separati dalla rete Internet a causa del gateway, i sensori sono comunque accessibili mediante IPv6 grazie al layer di adattamento 6LoWPAN [12, 13]. 6LoWPAN, cioè *IPv6 over Low power Wireless Personal Area Networks* è uno strato software sviluppato da un gruppo di lavoro dell'IETF, che adatta il protocollo IPv6, di per sé molto oneroso, all'utilizzo su dispositivi dalle limitate capacità di calcolo basati su IEEE 802.15.4 come i sensori. Le principali funzionalità di questo layer sono la compressione dell'header IP e la frammentazione dei pacchetti per rispettare la MTU (*Maximum Transmission Unit*) imposta da 802.15.4.

L'application server in Figura 1.2 è una macchina sulla quale viene eseguita l'applicazione Web WSNControl. Essa è per sua natura accessible attraverso qualunque dispositivo dotato di browser Internet, sia esso un computer, un telefono cellulare, un dispositivo embedded o quant'altro. Compito dell'applicazione Web è interrogare periodicamente le reti di sensori collegate a ciascun gateway conosciuto dall'applicazione, recuperarne i dati raccolti e visualizzarli in una qualche maniera significativa sul browser dell'utente.

A supporto della Web application è presente una posizione di archiviazione dei dati raccolti (storage), che può essere sfruttata per diversi scopi, come ad esempio l'applicazione della tecnica Compressive Sensing.

1.1 Architettura di WSNControl

Le funzionalità implementate dalla Web application sono: la creazione e modifica di testbed di sensori IEEE 802.15.4 attraverso un'interfaccia grafica, la visualizzazione dei dati raccolti dai sensori di uno o più testbed attraverso una rappresentazione tridimensionale, la ricostruzione dei pacchetti non ricevuti dai sensori a causa della scarsa qualità del mezzo di trasmissione wireless, ed infine la possibilità di interfacciarsi con l'applicazione attraverso qualunque tipo di dispositivo dotato di browser internet.

A partire da queste richieste di progetto, l'applicazione è stata sviluppata secondo la tipologia 3-tier, implementando cioè tre strati applicativi: il client tier, l'application tier e il database tier.

Il client tier è stato tenuto il più leggero e flessibile possibile, e viene eseguito da un browser Web; in questo modo si garantisce l'accesso all'applicazione di monitoraggio attraverso un'ampia gamma di dispositivi, siano essi Personal Computer, telefoni cellulari o PDA.

L'application tier è realizzato su piattaforma Java EE [14], ad eccezione della funzione di ricostruzione, implementata in C++, che offre migliori prestazioni. Poiché è in questo tier che ha luogo la maggior parte dell'elaborazione e dello scambio dei dati, è opportuno che tale componente software sia installato su una macchina dedicata, specialmente se le reti da monitorare sono di grandi dimensioni.

Infine il database tier contiene un database relazionale il cui scopo è fornire un solido ed efficiente archivio per i dati raccolti, nonché una potente interfaccia di accesso ai dati. Database tier ed application tier sono stati collocati sulla stessa macchina, ma possono altresì essere messi in funzione su macchine diverse.

A quanto finora descritto va ancora aggiunto il gateway, il quale funge da anello di congiunzione tra le reti di sensori e l'applicazione di monitoraggio. Il funzionamento di questo dispositivo è piuttosto semplice: l'applicazione Web chiede al gateway di recuperare le ultime misurazioni dei sensori ed il gateway inoltra la richiesta ai sensori (a cui è direttamente collegato) riunendo i dati che gli pervengono. Infine restituisce il tutto all'application server.

Come si può vedere in Figura 1.3 l'applicazione è suddivisa ad alto livello in quattro moduli: comunicazione, visualizzazione, ottimizzazione e feedback. Ad essi va aggiunto anche il Data Tier, incaricato di memorizzare in forma persistente i dati raccolti dai sensori.
Nel dettaglio le funzioni dei moduli sono:

comunicazione: permette lo scambio di informazioni tra Gateway e Data Tier, attraverso un protocollo di comunicazione implementato ad hoc;

visualizzazione: permette all'utente finale di visualizzare alcune semplici pagine per la gestione, la modifica e il monitoraggio delle reti di sensori;

ottimizzazione: permette di ricostruire i dati non pervenuti all'applicazione, mediante un semplice algoritmo di recupero;

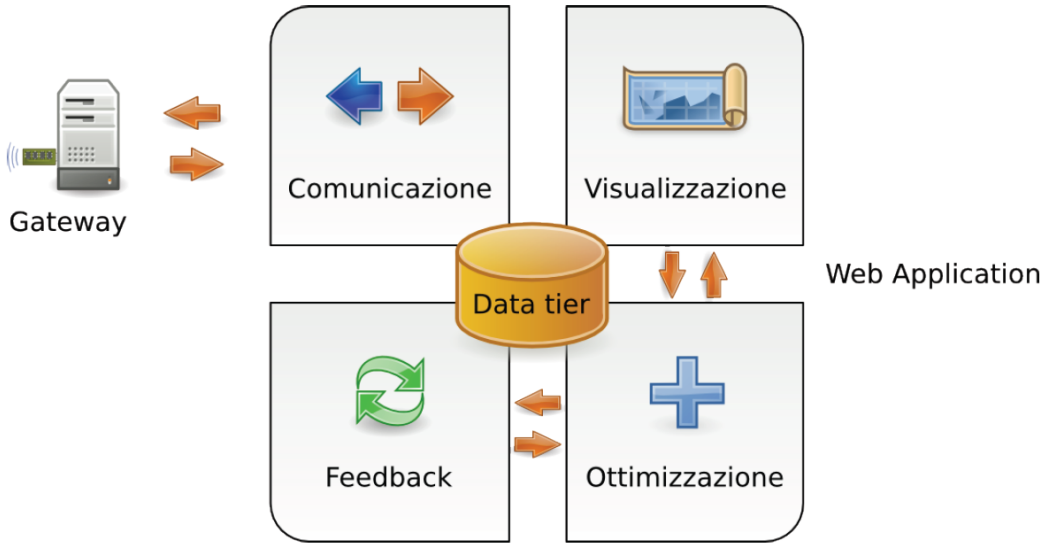


Figura 1.3: Architettura modulare dell'applicazione

feedback: è stato predisposto per l'integrazione della tecnica Compressive Sensing, come descritto nel Capitolo 5.

1.2 Data tier

Il Database tier, l'archivio a disposizione dell'applicazione, è stato realizzato secondo lo schema riportato in Tabella 1.1. Esso è composto da 4 tabelle con nomi autoesplicativi: la prima, *Sensors*, contiene un elenco di tutti i sensori gestiti dall'applicazione, qualunque sia la rete che li contiene. All'interno di ogni testbed ogni nodo è identificato da un attributo id. Ogni sensore nella tabella è allora identificato dalla coppia di campi id e testbed id che ne costituiscono la chiave primaria. Esiste poi un terzo campo testuale chiamato name attraverso il quale è possibile assegnare un nome mnemonico ai nodi.

La seconda tabella, chiamata *Testbeds*, contiene alcuni campi utili alla descrizione dei testbed associati all'applicazione. Così come avviene per i sensori, anche i testbed sono identificati in modo univoco tramite un apposito id, ma oltre a tale campo, nella tabella sono inclusi anche un campo di testo chiamato

Tabella	Campi
Sensors	id (int), name (varchar), testbed id (int)
Testbeds	id (int), name (varchar), url (text), gw host (text), gw port (int)
Types Of Measures	id (int), name (varchar)
Sensors Measures	id (int), sensor id (int), testbed id (int), TOM id (int), value (float), timestamp (bigint)

Tabella 1.1: Struttura del database

name, utile per recuperare il nome del testbed a partire dall'id e un campo url che contiene lo Uniform Resource Locator (URL) di un file XML descrittore del testbed. Poichè ogni testbed fa riferimento ad un gateway, sono stati aggiunti anche i campi gw host e gw port, grazie ai quali è possibile identificare quale dei gateway è responsabile di ogni testbed presente nella tabella.

La terza tabella, chiamata *Types Of Measures*, possiede i due campi id e name ed è stata pensata per associare un identificatore numerico univoco ad ogni tipologia di grandezza che un sensore può misurare.

La quarta tabella contiene invece una serie di misure riportate dal gateway all'applicazione. Tale tabella, chiamata Sensors Measures, contiene 6 campi ed è su questa tabella che vengono svolte tutte le operazioni di lettura e scrittura di misure. Il campo id identifica univocamente ogni misura ed è pertanto la chiave primaria. I due campi value e timestamp contengono il valore vero e proprio della misura, espresso come numero in virgola mobile e l'orario di acquisizione della stessa, espresso in formato UNIX Timestamp. Infine i campi sensor id, testbed id e TOM id contengono rispettivamente l'id del sensore, quello del testbed e il tipo di misura a cui si riferisce ciascuno dei valori; tramite essi è possibile filtrare le misure in base a tali identificatori.

1.3 Client tier

Il Client tier è la parte dell'applicazione che si occupa di istruire la web application con i comandi forniti dall'utente e restituire i risultati grazie ad una applet. L'interfaccia grafica consente di caricare una mappa di testbed precedentemente creata e di lavorare su di essa in due modalittà: una bidimensionale (o di modifica) ed una tridimensionale (o di monitoraggio).

La modalità di modifica consente di posizionare e muovere i sensori sulla mappa attraverso click e trascinamenti del mouse. Cliccando su un sensore con il tasto destro del mouse si può accedere ad un menù contestuale attraverso il quale è possibile ad esempio eliminarlo dalla mappa, o impostarne l'altezza dal terreno. Le informazioni su un sensore sono disponibili semplicemente tenendo fermo il mouse sopra ad esso. Attraverso il tasto Calibrate è inoltre possibile impostare la scala della mappa facendo click su due punti sulla mappa a distanza nota ed inserendo nella finestra che compare in seguito la reale distanza tra i due punti e l'unità di misura con cui è stata espressa.

La modalità di visualizzazione, visibile in Figura 1.4, consente di seguire in tempo reale l'evoluzione dei dati raccolti dai sensori di un testbed, in una visualizzazione tridimensionale che consente rotazioni, traslazioni e zoom sul grafico. Sul piano XY viene visualizzata la mappa, mentre delle barre con- cordi all'asse Z, posizionate in corrispondenza dei sensori della rete, riportano i valori misurati. L'altezza della barra è proporzionale al valore visualizzato. Attraverso il pannello View mode settings è possibile cambiare il tipo di misurazione da visualizzare nonchè l'intervallo di aggiornamento della mappa tridimensionale sottostante.

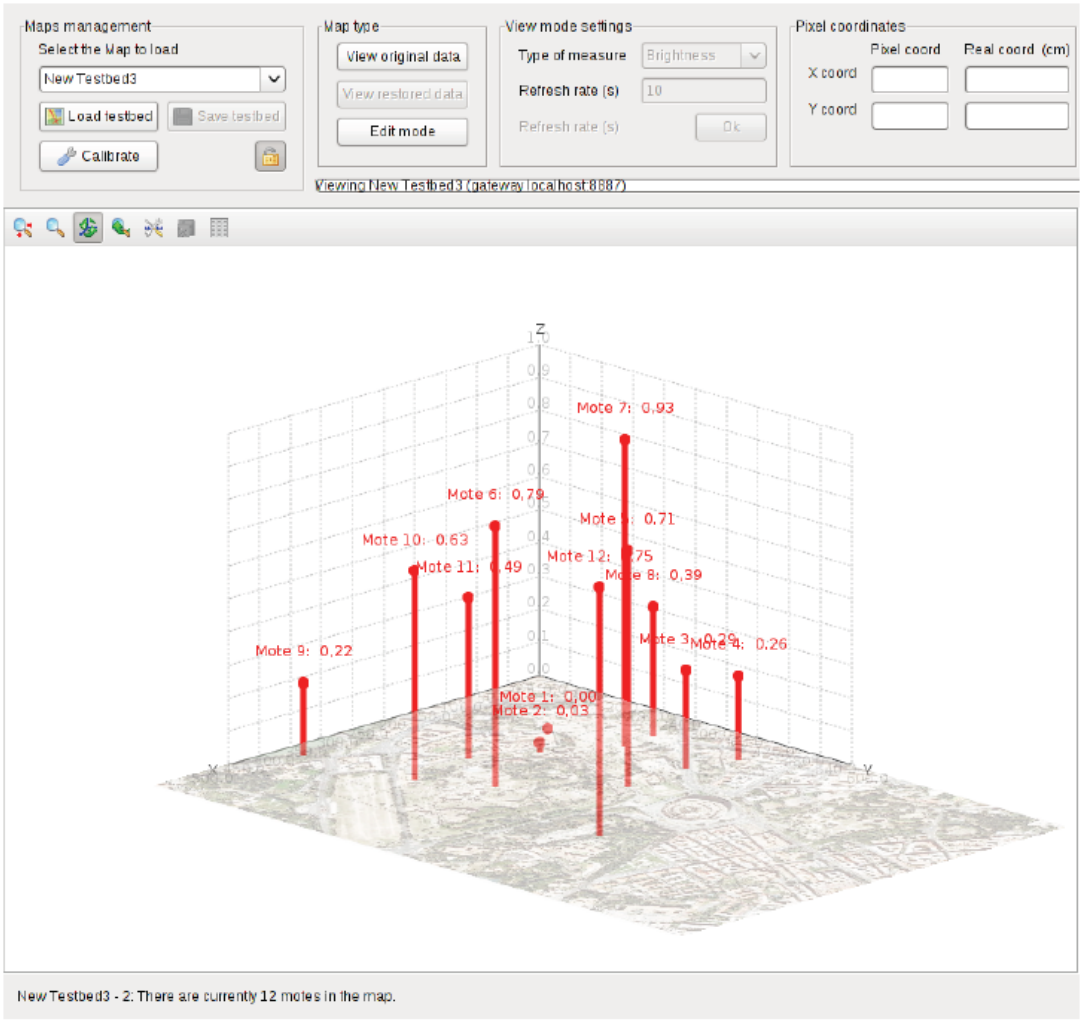


Figura 1.4: Screenshot della modalità di visualizzazione

1.3.1 Visualizzazione in HTML

L'applicazione offre la possibilità di visualizzare i dati raccolti dai sensori anche in modalita HTML, per cui è possibile interfacciarsi con l'applicazione attraverso un qualunque dispositivo provvisto di un browser. In questa modalità però non è possibile accedere alla modifica o alla creazione di una nuova rete. Anche dalla pagina di visualizzazione in HTML è possibile accedere alla visualizzazione tridimensionale. Quest'ultima riporta in formato immagine la medesima visualizzazione 3D disponibile nella applet, mentre nella parte sottostante è presente una tabella che riporta le misurazioni effettuate dai sensori sulla base del tipo di misura.

1.4 Application tier

L'application tier è quel componente dell'applicazione che esegue l'elaborazione delle richieste pervenute dal client. Dato il suo ruolo centrale nel corretto funzio-

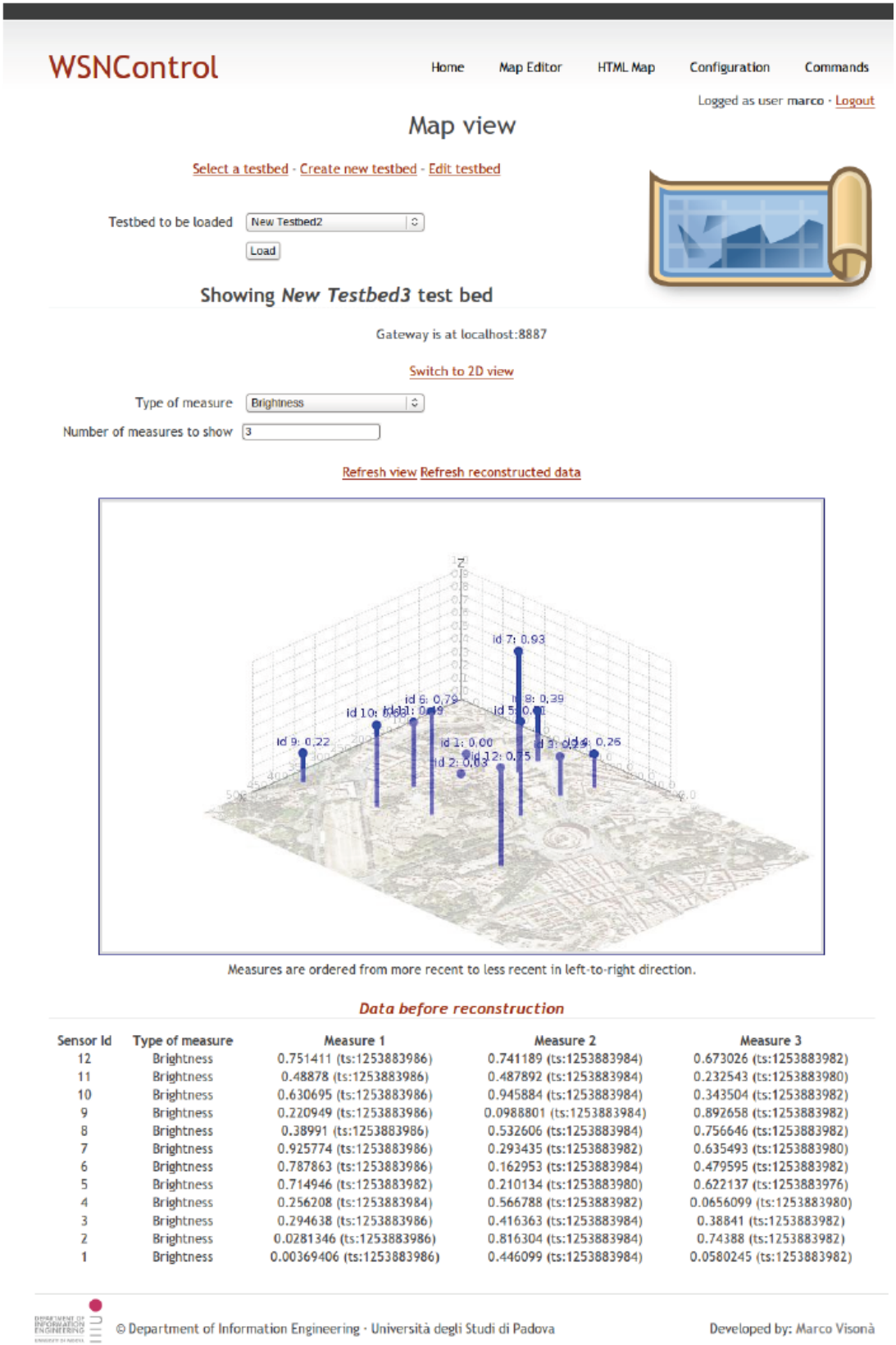


Figura 1.5: Screenshot della visualizzazione in modalità HTML

namento del sistema, tale componente è suddiviso in quattro moduli (Figua 1.3), ognuno dei quali si occupa di un aspetto peculiare del processing.

1.4.1 Modulo di comunicazione

Il modulo di comunicazione è quello che instaura la connessione verso i gateway delle reti di sensori, e ne richiede i dati a intervalli regolari. Per permettere una corretta comunicazione tra Web application e gateway è stato definito un semplice protocollo, chiamato *Web Application to Gateway Protocol* (WAGP). WAGP è un protocollo a livello applicativo che fa uso di TCP in modo da garantire la ricezione e la correttezza dei dati in transito. La comunicazione avviene in due fasi: nella prima il modulo di comunicazione invia un comando al gateway, se tale comando presuppone una risposta, si entra nella seconda fase, ovvero quella di ricezione dei dati.

La struttura del pacchetto contenente i dati è riportata in Figura 1.6. All'interno

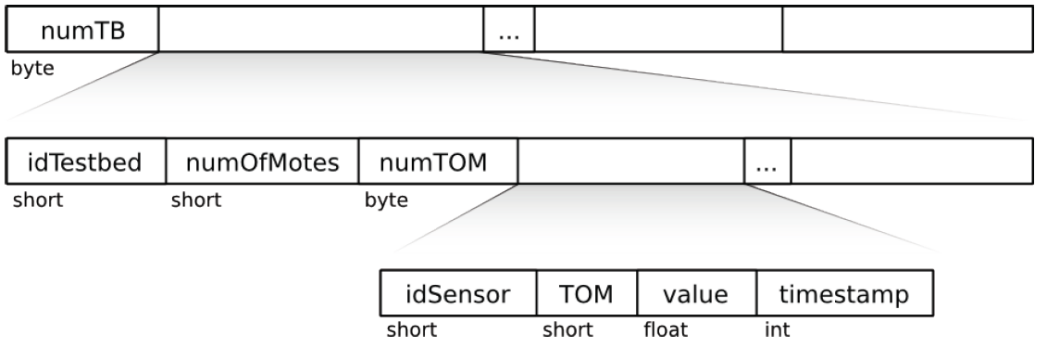


Figura 1.6: Struttura del pacchetto dati di WAGP.

delle caselle appare il nome del campo, mentre nell'area sottostante ne appare il tipo di dato. Il gateway può comunicare dati relativi a uno o più testbed, per questo come primo campo invia il numero di testbed di cui ha intenzione di inviare dati. In seguito seguono una serie di payload, uno per ogni testbed, al cui interno sono contenuti l'id del testbed corrente, il numero di nodi, il numero di tipi di misura supportati dai sensori che lo compongono ed infine, in coda a queste informazioni, vengono ripetuti i seguenti 4 campi: id del sensore, tipo di misura corrente, valore e timestamp della misura, in base ovviamente al numero di sensori e di tipi di misura per ogni sensore.

1.4.2 Modulo di visualizzazione

Il modulo di visualizzazione della Web application è un modulo situato in parte sul server ed in parte sul client che visualizza l'applicazione. Questo modulo contiene le classi che realizzano la applet di visualizzazione e modifica dei testbed, le servlet che forniscono i dati ai visualizzatori ed infine la directory radice della Web application in cui sono situate tutte le JavaServer Page.

1.4.3 Modulo di ottimizzazione e feedback

Nella prima realizzazione dell'applicazione WSNControl il modulo di ottimizzazione è stato implementato con il compito di ricostruire i dati non pervenuti all'applicazione. Si consideri un testbed formato da N nodi che, ad intervalli regolari, spediscono al gateway le loro ultime misurazioni. A causa della scarsa qualità del canale di comunicazione wireless, può accadere che alcuni pacchetti vengano persi e ne giungano al gateway soltanto K, con $K \leq N$. L'applicazione riceverà queste K misure ed applicherà loro un algoritmo di ricostruzione che permetta di ottenere nuovamente N misure, a partire dagli ultimi dati ricevuti e da quelli immagazzinati nel data tier. A questo scopo è stato implementato un semplice algoritmo di ricostruzione che riceve in ingresso i dati incompleti dell'ultima misurazione assieme alle precedenti m misure relative ad istanti precedenti e salvate nel database (con $1 < m < M$, dove M è la profondità scelta per il database, per ovvi motivi). Ogni misura è inoltre corredata dal rispettivo timestamp. L'algoritmo scandisce i dati in ingresso individua eventuali "buchi" nell'ultima lettura e li riempie con la misura più recente a disposizione nel database. Utilizzando questo approccio però non si ha alcun vantaggio dal punto di vista dell'efficienza della rete in quanto i dati che non giungono al gateway sono solo quelli persi per la scarsa qualità del canale; ai sensori viene comunque chiesto di trasmettere ad ogni round. L'obiettivo di questa tesi è invece l'introduzione di una tecnica che permetta di diminuire il numero di sensori attivi ad ogni round, cosicchè le risorse energetiche della rete possano essere preservate. Questa prima realizzazione del modulo di ottimizzazione è stata dunque modificata, per permettere non più il recupero dei pochi pacchetti persi durante la fase di raccolta dei dati, ma piuttosto la ricostruzione del segnale da una sua versione incompleta.

Nei prossimi capitoli verranno illustrate delle tecniche che permetteranno il recupero del segnale da una quantità minima di misure. Il modulo di ottimizzazione non verrà quindi utilizzato per sopperire alle mancanze della rete, ma piuttosto come strumento per diminuire il carico di lavoro della rete con conseguente risparmio di energia da parte dei sensori e aumento dell'efficienza. Assieme alla richiesta delle misure da parte del gateway infatti, ai sensori verrà comunicata una probabilità di trasmissione $p_{TX} = K/N$, cosicchè in media solo K sensori trasmetteranno la propria misura nel round di lettura corrente, le restanti verranno recuperate dall'algoritmo di ricostruzione. Nel Capitolo 5 in particolare saranno riportate le modifiche effettuate al modulo di ottimizzazione.

Il modulo di feedback è stato incluso nel progetto iniziale solo nominalmente, in vista dell'implementazione di un vero algoritmo di ricostruzione. Come verrà illustrato nei prossimi capitoli questo modulo ha il compito di stimare l'errore che si sta commettendo nel ricostruire il segnale e regolare di conseguenza la p_{TX} dell'iterazione successiva. I dettagli riguardanti le operazioni eseguite dal modulo verranno riportati nel Capitolo 5.

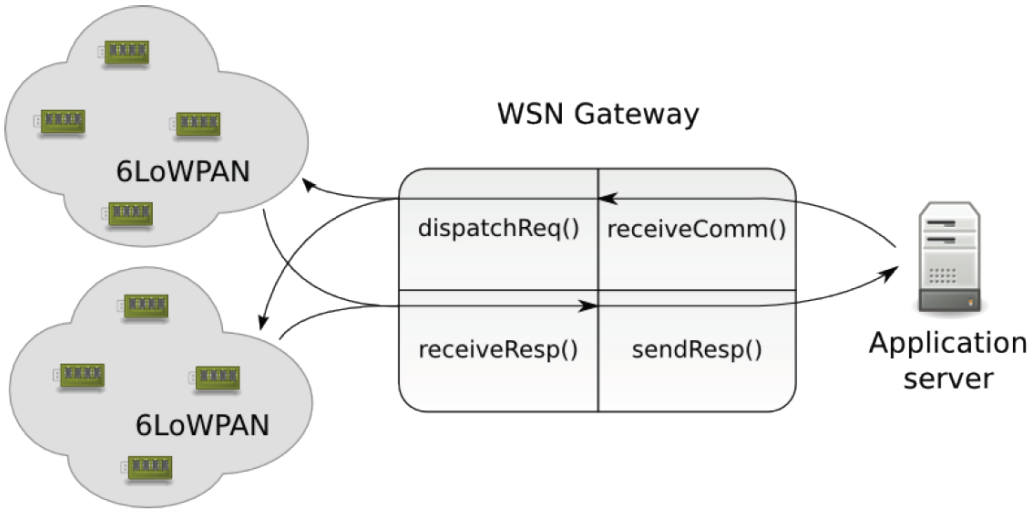


Figura 1.7: Architettura del funzionamento del Gateway

1.5 Gateway

Il gateway, nel sistema di monitoraggio descritto in Figura 1.2, è quel componente hardware e software che rende possibile la comunicazione tra i sensori e la Web application. Questo componente non è ancora stato implementato nel framework WSNControl. Tuttavia è noto comportamento che deve seguire ed è stata sviluppata una applicazione Java che lo simula. Ad un alto livello di astrazione, il gateway agisce restando in ascolto su una porta per una connessione TCP da parte dell'application server, il quale invia un comando sotto forma di intero ad 8 bit. Come si può osservare in Figura 1.7, il gateway è in grado di riconoscere il comando tramite una funzione `receiveComm()` e di inviare una serie di richieste al testbed appropriato attraverso la funzione `dispatchReq()`. A questo punto il gateway attende la risposta dalla rete che, a differenza della rete Internet attraverso cui comunica con l'application server, funziona su un link layer IEEE 802.15.4. I dati vengono ricevuti dalla funzione `receiveResp()` che effettua la conversione dal formato gestito dai sensori a quello di interscambio con l'applicazione. Il risultato viene quindi rispedito alla Web application dalla funzione `sendResp()`, nel formato indicato dal protocollo WAGP. In ogni caso il gateway è sostanzialmente un forwarder di pacchetti. Tutte le funzioni appena descritte sono state implementate in un programma Java chiamato *FakeGateway*. Il software attende connessioni in ingresso su una porta TCP, ad ogni richiesta da parte della web application risponde con un pacchetto contenente i dati dei sensori. I dati in questione vengono letti da alcuni file di testo che sono stati ricavati da alcune campagne di monitoraggio effettuate con la rete di sensori del dipartimento. Per le simulazioni si sono usati anche dati provenienti dalla rete di sensori *LUCE* (Lausanne Urban Canopy Experiment) [15].

2

Compressive Sensing e Principal Component Analysis

Nello scenario delle Reti di Sensori Wireless un problema chiave è quello della gestione e della compressione dei dati per incrementare l'efficienza della loro raccolta presso un server centrale (*sink*). Per risolvere questo problema, i dati vebgono compressi dei durante la fase di raccolta e vengono successivamente recuperati al sink. Compressive Sensing (CS) [4, 16] è una tecnica di compressione dei dati innovativa, che sfrutta la correlazione di un insieme di dati $\mathbf{x}$ per comprimerli mediante l'utilizzo di alcune matrici. Se la matrice di compressione e i dati originali $\mathbf{x}$ rispettano alcune proprietà, $\mathbf{x}$ può essere ricostruito a partire dalla sua versione compressa $\mathbf{y}$, grazie ad un algoritmo di ottimizzazione convessa.

2.1 Compressive Sensing

Si supponga di avere N sensori distribuiti casualmente in un ambiente che effettuano misure di un qualche tipo e si rappresentino i dati raccolti in un determinato istante mediante un vettore $\mathbf{x} \in \mathbb{R}^N$, dove N è la lunghezza del vettore. Al punto di raccolta dei dati, si vuole ricostruire il vettore $\mathbf{x}$ il più fedelmete possibile, ricevendo però solo una parte delle letture dei sensori, rappresentabile nel vettore $\mathbf{y} \in \mathbb{R}^M$ con $M < N$. Dal punto di vista della rete, il sink invia ad ogni sensore un dato che rappresenta la probabilità di trasmissione $p_{TX} = M/N$. Ogni sensore, in base a questa probabilità decide se rispondere alla richiesta inviando la propria lettura o meno, quindi in media solo M sensori invieranno dati al sink. La relazione che intercorre tra i due vettori può essere espressa mediante l'equazione:

$$\mathbf{y} = \mathbf{\Phi x} \,, \tag{2.1}$$

dove $\mathbf{\Phi} \in \mathbb{R}^{M \times N}$ è la matrice di routing, ovvero la matrice che descrive in che modo ogni elemento di $\mathbf{y}$ è legato a $\mathbf{x}$. Una particolare matrice $\mathbf{\Phi}$, quella che verrà usata per i nostri scopi, è la matrice che descrive il campionamento del vettore $\mathbf{x}$, cioè una matrice che ha un solo elemento di valore unitario per ogni riga e al più un 1 per colonna.

$$\mathbf{\Phi} = \begin{pmatrix} 0 & 1 & 0 & 0 & \cdots & 0 \\ 1 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & \cdots & 1 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 1 & \cdots & 0 \end{pmatrix} .$$

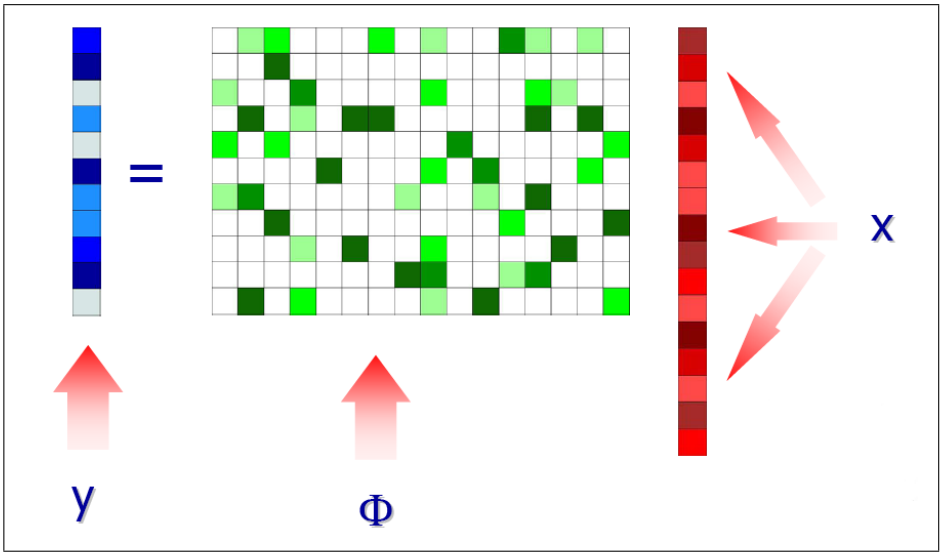


Figura 2.1: Rappresentazione concettuale dell'equazione (2.1)

Si supponga inoltre che esista una trasformazione invertibile attraverso la quale il vettore $\mathbf{x}$ possa essere reso sparso. In dettaglio, deve esistere una matrice $\mathbf{\Psi}$ invertibile e ortonormale, ovvero tale che $\mathbf{\Psi}\mathbf{\Psi}^{\star} = \mathbf{I}$, di dimensioni N $\times$ N tale per cui:

$$\mathbf{x} = \mathbf{\Psi}\mathbf{s} \, , \tag{2.2}$$

e $\mathbf{s}$ risulta essere K–sparso, ossia definito in questo modo:

Definizione 2.1. *Vettore K–sparso*
Sia $\mathbf{v} \in \mathbb{R}^N$ un vettore con N componenti. Si dice che $\mathbf{V}$ è un ***vettore K–sparso*** *se soltanto K delle sue N componenti sono diverse da zero.*

□

Questa ipotesi è molto forte, e in generale non è detto che la trasformazione esista, nella sezione 2.2 verrà presentato un metodo per trovare una matrice $\mathbf{\Psi}$ a partire dallo studio della statistica del segnale, chiamato Principal Component Analysis.

Ora, combinando (1) e (2) si può scrivere:

$$\mathbf{y} = \mathbf{\Phi}\mathbf{x} = \mathbf{\Phi}\mathbf{\Psi}\mathbf{s} = \mathbf{A}\mathbf{s} \, . \tag{2.3}$$

Al sink si hanno a disposizione il vettore $\mathbf{y}$ e la matrice $\mathbf{A}$ e si cerca di ricostruire il vettore $\mathbf{s}$. Una volta ricostruito $\mathbf{s}$ è possibile ottenere $\mathbf{x}$ attraverso (2.2).

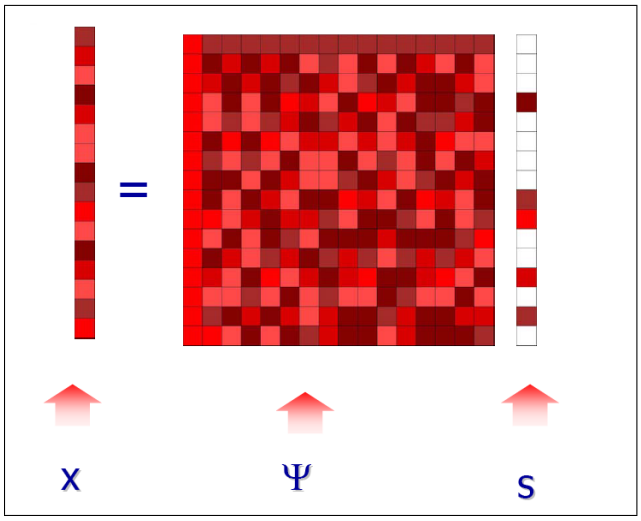


Figura 2.2: Rappresentazione concettuale dell'equazione (2.2), le celle di colore bianco rappresentano elementi nulli o trascurabili

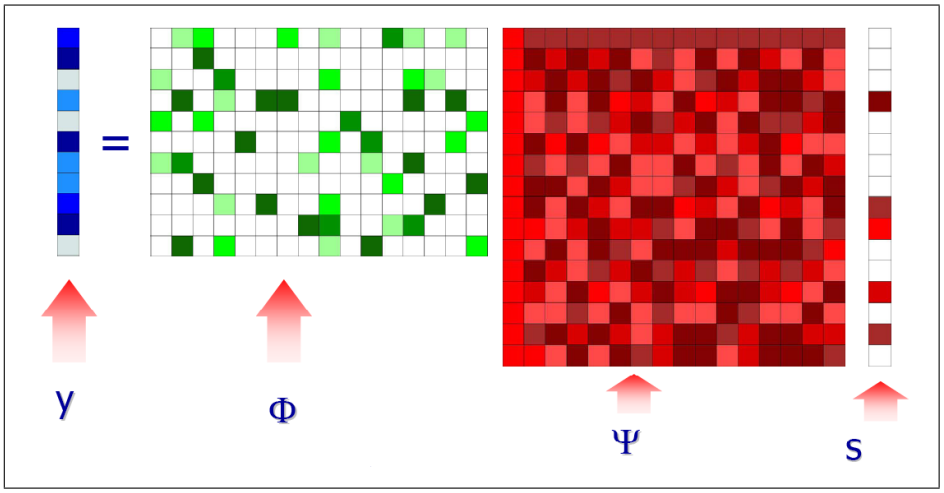


Figura 2.3: Rappresentazione concettuale dell'equazione (2.3)

Il sistema descritto da (2.3) è un sistema di N equazioni in M incognite e in generale ammette uno spazio di soluzioni $N-M$ dimensionale. Ogni elemento di questo spazio può essere usato per ricostruire $\mathbf{x}$, ma si è interessati alla soluzione $\hat{\mathbf{s}}$ che minimizza l'errore di ricostruzione, definito come segue:

Definizione 2.2. *Errore di ricostruzione*
Sia $\mathbf{x} \in \mathbb{R}^N$ *il vettore contenente le misure di una rete di* N *sensori e sia* $\hat{\mathbf{x}}$ *un vettore ottenuto dalla ricostruzione del vettore* $\mathbf{x}$ *basata su CS. Si definisce* ***errore di ricostruzione****, e si indica con* ξ_R*, la quantità:*

$$\xi_R = \frac{\|\mathbf{x} - \hat{\mathbf{x}}\|_{\ell_2}}{\|\mathbf{x}\|_{\ell_2}} \,. \tag{2.4}$$

□

Dove la funzione $\|\cdot\|_{\ell_2}$ indica la norma ℓ_2, definita come segue:

Definizione 2.3. *Norma ℓ_2*
Sia $\mathbf{x} \in \mathbb{R}^N$ un vettore, e siano x_i con $i \in \{1, \ldots, N\}$ le sue componenti. Si definisce ***norma*** *ℓ_2 di $\mathbf{x}$, e si indica con $\|\mathbf{x}\|_{\ell_2}$ la quantità:*

$$\|\mathbf{x}\|_{\ell_2} = \left(\sum_{i=1}^{N} x_i^2 \right)^{\frac{1}{2}} . \tag{2.5}$$

□

É stato dimostrato [17, 18] che, dato il vettore K–sparso $\mathbf{s}$, se per ogni altro vettore K–sparso $\mathbf{v}$, con lo stesso numero di elementi diversi da zero di $\mathbf{s}$, e per qualche $\epsilon > 0$ vale:

$$1 - \epsilon \leq \frac{\|\mathbf{A}\mathbf{v}\|_{\ell_2}}{\|\mathbf{v}\|_{\ell_2}} \leq 1 + \epsilon \, , \tag{2.6}$$

ovvero la matrice $\mathbf{A}$ preserva la lunghezza dei vettori K–sparsi, allora la soluzione esiste, è unica ed è il vettore $\mathbf{s}$ che soddisfa il sistema (2.3) con il minor numero di elementi non nulli, o equivalentemente, il più sparso. Una condizione sufficiente affinche la soluzione sia unica è che la matrice $\mathbf{A}$ soddisfi la (2.6) per un generico vettore $3K$–sparso. Questa condizione viene chiamata *restricted isometry property* (RIP). Un'altra condizione, equivalente a quest'ultima, è l'incoerenza delle matrici $\mathbf{\Phi}$ e $\mathbf{\Psi}$. Si definisca come segue la coerenza tra due matrici:

Definizione 2.4. *Coerenza tra due matrici*
Siano $\mathbf{\Phi}$ e $\mathbf{\Psi}$ due matrici. Indicando con $\{\phi_k\}$ le righe della matrice $\mathbf{\Phi}$ e con $\{\psi_j\}$ le colonne di $\mathbf{\Psi}$, che devono avere la lunghezza N, si definisce ***coerenza*** *tra le matrici $\mathbf{\Phi}$ e $\mathbf{\Psi}$ la quantità:*

$$\mu(\mathbf{\Phi}, \mathbf{\Psi}) = \sqrt{N} \cdot \max_{1 \leq k,j \leq N} |\langle \{\phi_k\}, \{\psi_j\} \rangle| \ . \tag{2.7}$$

□

In pratica questa quantità misura la massima correlazione tra gli elementi delle due matrici. Se gli elementi sono molto correlati la coerenza è alta. Segue da alcuni sviluppi algebrici che $\mu(\mathbf{\Phi}, \mathbf{\Psi}) \in [1, \sqrt{N}]$ [4]. La condizione richiede dunque che la massima correlazione tra gli elementi di $\mathbf{\Phi}$ e $\mathbf{\Psi}$ sia piccola , o equivalentemente, che non sia possibile rappresentare le righe $\{\phi_k\}$ di $\mathbf{\Phi}$ tramite una combinazione lineare sparsa delle colonne $\{\psi_j\}$ di $\mathbf{\Psi}$ e viceversa [18]. Usando una matrice $\mathbf{\Phi}$ ottenuta da un campionamento casuale sia la RIP che l'incoerenza con $\mathbf{\Psi}$ sono soddisfatte con alta probabilità.

L'algoritmo di ricostruzione del segnale dunque deve ricavare gli N campioni del segnale $\mathbf{x}$, a partire dagli M campioni di $\mathbf{y}$ e dalla matrice $\mathbf{A} = \mathbf{\Phi}\mathbf{\Psi}$. Come già detto le soluzioni del sistema (2.3) sono infinite, e sono tutte contenute nello spazio $N - M$ dimensionale $\mathcal{H} = \mathcal{N}(\mathbf{A}) + \mathbf{s}$, dove $\mathcal{N}(\mathbf{A})$ indica il nucleo di $\mathbf{A}$.

L'approccio più semplice a un problema di questo tipo è quella di trovare la soluzione con norma ℓ_2 minore nel nucleo di $\mathbf{A}$ traslato in $\mathbf{s}$ risolvendo:

$$\begin{aligned} \hat{\mathbf{s}} &= \arg\min \|\mathbf{s}\|_{\ell_2} \ ; \\ &\text{con} \quad \mathbf{y} = \mathbf{A}\mathbf{s} \ . \end{aligned} \tag{2.8}$$

Questa ottimizzazione ha una forma chiusa semplice, ma purtroppo la soluzione trovata non è quasi mai sparsa.

Vengono ora definite le due funzioni $\|\cdot\|_{\ell_0}$ e $\|\cdot\|_{\ell_1}$:

Definizione 2.5. *Norma ℓ_0*
*Sia $\mathbf{x} \in \mathbb{R}^N$ un vettore, e siano x_i con $i \in \{1, \dots, N\}$ le sue componenti. Si definisce **norma** ℓ_0 di $\mathbf{x}$, e si indica con $\|\mathbf{x}\|_{\ell_0}$ la quantità:*

$$\|\mathbf{x}\|_{\ell_0} = \sum_{i=1}^{N} \mathbf{1}\{x_i \neq 0\} \ , \tag{2.9}$$

dove $\mathbf{1}\{\cdot\}$ indica la funzione indicatice.

□

Definizione 2.6. *Norma ℓ_1*
*Sia $\mathbf{x} \in \mathbb{R}^N$ un vettore, e siano x_i con $i \in \{1, \dots, N\}$ le sue componenti. Si definisce **norma** ℓ_1 di $\mathbf{x}$, e si indica con $\|\mathbf{x}\|_{\ell_1}$ la quantità:*

$$\|\mathbf{x}\|_{\ell_1} = \sum_{i=1}^{N} |x_i| \ . \tag{2.10}$$

□

Risolvendo il problema di minimizzazione sostituendo alla norma ℓ_2 la norma ℓ_0, che conta il numero di elementi non nulli, si ottiene la soluzione più sparsa, cioè quella con il maggior numero di zeri:

$$\begin{aligned} \hat{\mathbf{s}} = \arg\min \|\mathbf{s}\|_{\boldsymbol{\ell_0}} \ ; \\ \text{con} \quad \mathbf{y} = \mathbf{A}\mathbf{s} \ . \end{aligned} \tag{2.11}$$

Sfortunatamente il problema (2.12) è un problema NP-completo e numericamente instabile. Per problemi di grande dimensione è quindi computazionalmente impossibile da risolvere.

La soluzione è quella di utilizzare un algoritmo di minimizzazione della norma ℓ_1 che risolva il problema:

$$\begin{aligned} \hat{\mathbf{s}} = \arg\min \|\mathbf{s}\|_{\boldsymbol{\ell_1}} \ ; \\ \text{con} \quad \mathbf{y} = \mathbf{A}\mathbf{s} \ . \end{aligned} \tag{2.12}$$

Se la matrice $\mathbf{A}$ soddisfa la RIP, la soluzione del problema (2.13) è la stessa del problema (2.12), ma il costo computazionale è trattabile, anche per problemi di grande dimensione.

Per capire il motivo per cui la minimizzazione in norma ℓ_2 non porta ad una soluzione sparsa si consideri lo spazio $\mathbb{R}^3$ mostrato in Figura 2.4. L'insieme di tutti i vettori sparsi in $\mathbb{R}^N$ si trova negli iperpiani allineati agli assi, come mostrato in Figura 2.4(a) nel caso tridimensionale. Si indichi con $\mathbf{s}$ la soluzione ottima del problema e con $\mathcal{H}$ al nucleo di $\mathbf{A}$ traslato in $\mathbf{s}$ ($\mathcal{H}$ è orientato in modo casuale

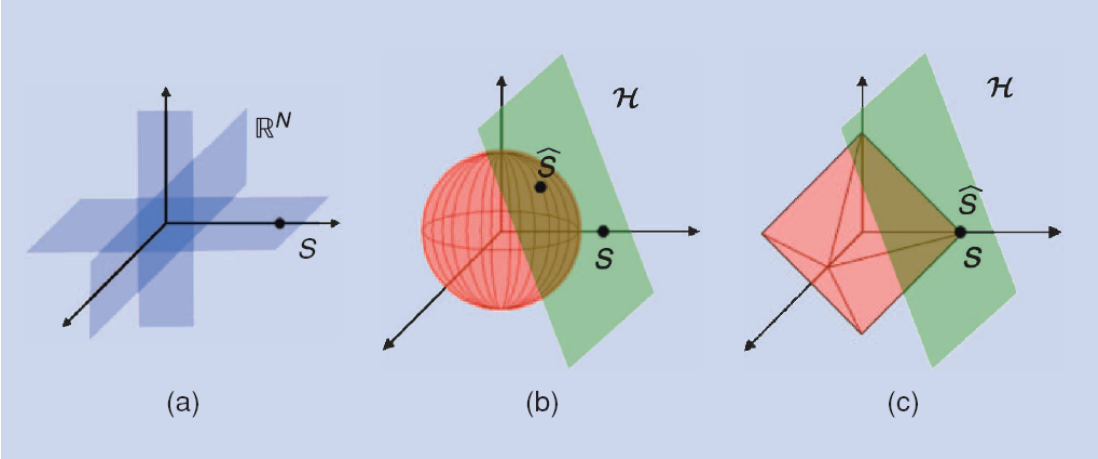


Figura 2.4: (a) I sottospazi contenenti i vettori sparsi in $\mathbb{R}^3$ stanno lungo gli assi; (b) Visualizzazione del punto di contatto $\hat{\mathbf{s}}$ tra la palla ℓ_2 e il piano $\mathcal{H}$; (c) Visualizzazione del punto di contatto tra la palla ℓ_1 e il piano $\mathcal{H}$.

perchè $\mathbf{\Phi}$ è una matrice di elementi casuali, in questo caso). La soluzione minima $\hat{\mathbf{s}}$ in norma ℓ_2 è il punto di $\mathcal{H}$ più vicino all'origine, per trovarlo si può espandere un'ipersfera (una palla ℓ_2) centrata nell'origine fino a toccare un punto di $\mathcal{H}$, come mostrato in Figura 2.4(b). Con alta probabilità $\hat{\mathbf{s}}$ è lontano da $\mathbf{s}$ e non è sparso. Al contrario se si espande una palla ℓ_1, come mostrato in Figura 2.4(c), il punto $\hat{\mathbf{s}}$ sarà in corrispondenza della soluzione sparsa $\mathbf{s}$.

Al sink è dunque necessaria l'implementazione di un metodo che ricerca la soluzione con norma ℓ_1 minore tra le possibili soluzioni di (2.3).

L'ipotesi di esistenza della matrice $\mathbf{\Psi}$ è un'ipotesi molto forte e non sempre verificata quando si sta lavorando con segnali reali. Usando alcune trasformazioni note, come la Discrete Cosine Transform, o la Haar Wavelet infatti non si ottengono vettori abbastanza sparsi [5]. Questo problema può essere risolto utilizzando un metodo statistico, la Principal Component Analysis, che sfrutta la correlazione spaziale e temporale dei dati $\mathbf{x}$.

2.2 Compressive Sensing e Principal Component Analysis

La Principal Component Analysis (PCA) è una procedura matematica che permette di rappresentare un insieme di variabili, eventualmente correlate, con un minor numero di variabili non correlate chiamate componenti principali. Per fare ciò è richiesta la conoscenza della statistica del primo ordine delle variabili di partenza. A seconda del campo di applicazione, la PCA è anche chiamata Karhunen-Loève Transform, Hotelling Transform o Proper Orthogonal Decomposition. In questo contesto useremo la PCA per costruire una matrice di trasformazione che renda sparso un segnale N-dimensionale in un qualche dominio. Il risultato teorico che sta alla base della PCA è il *teorema di Ky Fan*.

Teorema 1. *Teorema di Ky Fan*
Sia $\mathbf{\Sigma} \in \mathbb{R}^{N\times N}$ *una matrice simmetrica, siano* $\lambda_1 \geq \cdots \geq \lambda_N$ *i suoi autovalori e* $\mathbf{u_1}, ..., \mathbf{u_N}$ *gli autovettori associati (che senza perdita di generalità vengono assunti ortonormali). Dati M vettori ortonormali* $\mathbf{b_1}, ..., \mathbf{b_M}$ *in* $\mathbb{R}^N$*, con* $M \leq N$*, vale che:*

$$\max_{b_1,...,b_M} \sum_{j=1}^{M} \mathbf{b_j^T \Sigma b_j} = \sum_{i=1}^{M} \lambda_i \ , \tag{2.13}$$

e il massimo si ottiene quando $\mathbf{b_i} = \mathbf{u_i}, \forall i$.

□

Si definisca il vettore $\mathbf{x}^{(k)} \in \mathbb{R}^N$ come il vettore delle letture degli N sensori al tempo k. Si supponga di collezionare i dati relativi a tutti i sensori per K istanti di tempo successivi e di calcolare il vettore delle medie $\overline{\mathbf{x}}$ e la matrice delle covarianze $\widehat{\mathbf{\Sigma}}$ su questo insieme di dati.

$$\overline{\mathbf{x}} = \frac{1}{K}\sum_{k=1}^{K} \mathbf{x}^{(k)} \ , \ \widehat{\mathbf{\Sigma}} = \frac{1}{K}\sum_{k=1}^{K} (\mathbf{x}^{(k)} - \overline{\mathbf{x}})(\mathbf{x}^{(k)} - \overline{\mathbf{x}})^T \ . \tag{2.14}$$

Da un punto di vista geometrico si può considerare $\mathbf{x}^{(k)}$ come un punto in uno spazio a N dimensioni e cercare il piano M-dimensionale più vicino a $\mathbf{x}^{(k)}$. Come afferma il teorema di Ky Fan, la trasformazione lineare che conserva maggiormente l'informazione contenuta nel segnale $\mathbf{x}^{(k)}$ si può costruire dagli autovettori della matrice di covarianza. Se si definisce $\mathbf{U_M}$ come la matrice formata dagli M autovettori corrispondenti agli M autovalori maggiori di $\widehat{\mathbf{\Sigma}}$, la proiezione M dimensionale di $\mathbf{x}^{(k)}$ si può scrivere come:

$$\widehat{\mathbf{x}^{(k)}} = \overline{\mathbf{x}} + \mathbf{U_M}{\mathbf{U_M}}^T(\mathbf{x}^{(k)} - \overline{\mathbf{x}}) \ . \tag{2.15}$$

Se si definisce $\mathbf{s} = \mathbf{s_N} = {\mathbf{U_N}}^T(\mathbf{x}^{(k)} - \overline{\mathbf{x}})$, si può notare che per come è costruita la matrice ${\mathbf{U_N}}^T$, gli elementi del vettore $\mathbf{s}$ risultano ordinati in modo non crescente. Se per $i > M$ si ha che $s(i) \ll s(j)$, con $j = 1, \ldots, M$, allora $\mathbf{x}^{(k)}$ è ben approssimato da (2.15) attraverso l'utilizzo di $M < N$ coefficienti, cioè quasi tutta l'energia è concentrata nelle prime componenti. Il che equivale a dire che il punto $\mathbf{x}^{(k)} \in \mathbb{R}^N$ viene trasformato nel punto $\mathbf{s_M} \in \mathbb{R}^M$ in questo modo:

$$\mathbf{s_M} = {\mathbf{U_M}}^T(\mathbf{x}^{(k)} - \overline{\mathbf{x}}) \ . \tag{2.16}$$

Il risultato è che $\mathbf{s} = \begin{bmatrix} \mathbf{s_M} \\ \mathbf{0_{N-M}} \end{bmatrix}$ è un vettore M-sparso. Inoltre la matrice $\mathbf{U_N}$ è ortonormale, e quindi $\mathbf{U_N}{\mathbf{U_N}}^T = \mathbf{I_N}$ che è la matrice identità. Si è quindi in possesso di una trasformazione che rende sparso il segnale d'ingresso $\mathbf{x}^{(k)}$ in un altro dominio, inoltre questa trasformazione è invertibile e ortonormale. Sulla base dei risultati visti in precedenza è immediato verificare come si possa utilizzare la matrice $\mathbf{U_N}$ come matrice di trasformazione. Si ha infatti:

$$\mathbf{x}^{(k)} - \overline{\mathbf{x}} = \mathbf{U_N}\mathbf{s}^{(k)} = \mathbf{\Psi}\mathbf{s}^{(k)} \ , \tag{2.17}$$

e ricordando che $\mathbf{y}^{(k)} = \mathbf{\Phi}\mathbf{x}^{(k)}$ si può scrivere:

$$\mathbf{y}^{(k)} - \mathbf{\Phi}\overline{\mathbf{x}} = \mathbf{\Phi}(\mathbf{x}^{(k)} - \overline{\mathbf{x}}) = \mathbf{\Phi}\mathbf{U_N}\mathbf{s}^{(k)} \ , \tag{2.18}$$

la cui forma è simile a quella vista nell'equazione (2.3). Il recupero di $\mathbf{x}^{(k)}$ avviene dunque a partire da $\mathbf{y}$ e da $\mathbf{A} = \mathbf{\Phi}\mathbf{U_N}$, cercando, come accadeva nel CS, la soluzione $\hat{\mathbf{s}}^{(k)}$ con norma ℓ_1 minore, che soddisfa il sistema (2.18). $\hat{\mathbf{s}}^{(k)}$ sarà un vettore sparso, con solo M componenti non nulle. Una volta ottenuto $\hat{\mathbf{s}}^{(k)}$ il recupero del segnale avviene applicando la seguente trasformazione:

$$\hat{\mathbf{x}}^{(k)} = \overline{\mathbf{x}} + \mathbf{U_N}\hat{\mathbf{s}}^{(k)} \ . \tag{2.19}$$

In questo modo si ottiene un metodo, che definiamo CS+PCA, il quale sfrutta la matrice di trasformazione calcolata attraverso la Principal Component Analysis e i risultati teorici della tecnica Compressive Sensing per poter ricostruire il segnale $\mathbf{x}^{(k)}$ a partire dalla sua versione campionata $\mathbf{y}^{(k)}$. Usando il metodo CS+PCA si ottengono risultati migliori (in termini di errore di ricostruzione) rispetto a quanto si ottiene con CS affiancato da trasformate note [19].

A questo punto si possono distinguere due metodi con i quali applicare il metodo CS+PCA:

- Il primo metodo prevede l'alternarsi di due regimi di funzionamento della rete. Durante il primo regime tutti i sensori trasmettono con probabilità $p_{TX} = 1$ e il sink colleziona tutti i dati e li utilizza per costruire le statistiche necessarie. Successivamente la probabilità di trasmissione viene ridotta ($p_{TX} < 1$) e il sink applica la tecnica appena vista per ricostruire il segnale completo. In questo modo le statistiche vengono rinnovate di tanto in tanto e usate per gli istanti successivi. Questa tecnica sarà usata nel Capitolo 4, per testare le prestazioni dell'algoritmo di Nesterov, trattato nel Capitolo 3.

- Il secondo metodo prevede una fase iniziale in cui si studiano le statistiche del segnale durante K round in cui vengono collezionati i dati di tutti i sensori ($p_{TX} = 1$). Dopo l'inizializzazione le statistiche vengono aggiornate utilizzando di volta in volta il segnale ricostruito. Ad ogni round di questa fase la probabilità con cui i sensori trasmettono è $p_{TX} \leq 1$ e al sink vengono ricalcolate le statistiche sugli ultimi K campioni di $\mathbf{x}_k$. Usando questo metodo si può inoltre introdurre un controllo della qualità del segnale ricostruito che va ad agire sulla probabilità di trasmissione p_{TX} dei sensori in modo dinamico. Questa sarà la tecnica implementata nel modulo di ricostruzione trattato nel Capitolo 5.

3

Algoritmo di Nesterov

In questo capitolo si affronta lo studio di un algoritmo di programmazione convessa chiamato algoritmo di Nesterov. Per comprendere il funzionamento di tale algoritmo nella Sezione 3.1 vengono riportate alcune definizioni utili, successivamente vengono illustrati il metodo del gradiente (Sezione 3.2), il metodo dei moltiplicatori di Lagrange (Sezione 3.3) e le condizioni di Karush–Kuhn–Tucker (Sezione 3.4) che ne permettono l'estensione. Tutti questi concetti teorici sono necessari per analizzare le operazioni svolte dall'algoritmo di Nesterov, che viene riportato e descritto in Sezione 3.5. Nella Sezione 3.6 viene invece trattato un modo per applicare questo metodo al problema del Compressive Sensing, chiamato NESTA.

3.1 Definizioni

Definizione 3.1. *Punti di massimo e minimo assoluti*
Sia $f : X \to \mathbb{R}$ una funzione a valori reali definita su un insieme $X \subseteq \mathbb{R}^n$ e sia $\mathbf{x}_0 \in X$ un punto del dominio di f. Si dice che $\mathbf{x}_0$ è un ***punto di massimo assoluto*** *se per ogni $\mathbf{x} \in X$ tale che $\mathbf{x} \neq \mathbf{x}_0$ si ha:*

$$f(\mathbf{x}) \leq f(\mathbf{x}_0) \ .$$

Si dice che $\mathbf{x}_0$ è un ***punto di minimo assoluto*** *se per ogni $\mathbf{x} \in X$ tale che $\mathbf{x} \neq \mathbf{x}_0$ si ha:*

$$f(\mathbf{x}) \geq f(\mathbf{x}_0) \ .$$

Inoltre, i punti di massimo e minimo assoluti si dicono ***propri*** *se le rispettive condizioni sono verificate in senso stretto (senza la condizione di uguaglianza).*

□

Definizione 3.2. *Punti di massimo e minimo relativi*
Sia $f : X \to \mathbb{R}$ una funzione a valori reali definita su un insieme $X \subseteq \mathbb{R}^n$ e sia $\mathbf{x}_0 \in X$ un punto del dominio di f. Si dice che $\mathbf{x}_0$ è un ***punto di massimo relativo*** *se esiste un opportuno intorno $B(\mathbf{x}_0)$ di $\mathbf{x}_0$ tale che per ogni $\mathbf{x} \in B(\mathbf{x}_0)$ si ha:*

$$f(\mathbf{x}) \leq f(\mathbf{x}_0) \ .$$

Si dice che $\mathbf{x}_0$ *è un* ***punto di minimo relativo*** *se per ogni* $\mathbf{x} \in B(\mathbf{x}_0)$ *si ha:*

$$f(\mathbf{x}) \geq f(\mathbf{x}_0) \ .$$

□

Definizione 3.3. *Massimo assoluto, minimo assoluto, massimo relativo, minimo relativo*
Il valore che assume f *in un punto di massimo assoluto, di minimo assoluto, di massimo relativo, di minimo relativo si dice rispettivamente* ***massimo assoluto, minimo assoluto, massimo relativo, minimo relativo*** *di* f *in* X*.*

□

Definizione 3.4. *Gradiente di una funzione*
Sia f *una funzione di classe* $C^1(X)$[1] *con* X *insieme aperto* $X \subseteq \mathbb{R}^n$*. Si dice* ***gradiente di*** $\mathbf{f}$*, e si indica con* ∇f*, il vettore riga così definito:*

$$\nabla f \triangleq \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n} \right) \ .$$

□

Definizione 3.5. *Punto critico*
Sia $f : X \to \mathbb{R}$ *una funzione a valori reali definita su un insieme aperto* $X \subseteq \mathbb{R}^n$ *e sia* $\mathbf{x}_0 \in X$ *un punto del dominio di* f*. Si dice che* $\mathbf{x}_0$ *è un* ***punto critico*** *se tutte le derivate parziali calcolate in* $\mathbf{x}_0$ *sono nulle, ossia:*

$$\frac{\partial f(\mathbf{x}_0)}{\partial x_1} = 0 \quad , \quad \frac{\partial f(\mathbf{x}_0)}{\partial x_2} = 0 \quad , \ldots , \quad \frac{\partial f(\mathbf{x}_0)}{\partial x_n} = 0 \ ,$$

o, analogamente:

$$\nabla f(\mathbf{x}_0) = (0, 0, \ldots, 0) \ .$$

□

Definizione 3.6. *Matrice jacobiana e determinante jacobiano*
Date m *funzioni* $f_1(\mathbf{x}), \ldots, f_m(\mathbf{x})$ *definite in* $\mathbb{R}^n$*, si dice* ***matrice jacobiana*** *associata alle* m *funzioni la matrice* $m \times n$ *le cui righe sono i gradienti di queste, ossia:*

$$\mathbf{J}(\mathbf{x}) \triangleq \begin{bmatrix} \nabla f_1(\mathbf{x})^T \\ \nabla f_2(\mathbf{x})^T \\ \vdots \\ \nabla f_m(\mathbf{x})^T \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial x_2} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix} \ .$$

[1] L'appartenenza alla classe $C^1(X)$ indica una che la funzione è continua in X, derivabile e con derivata prima continua.

Se $n = m$ la matrice jacobiana risulta quadrata, allora si dice ***determinante jacobiano*** *o semplicemente jacobiano delle funzioni $f_1, \ldots, f_m$ il determinante della matrice jacobiana associata a queste, ossia:*

$$\frac{\partial(f_1, \ldots, f_m)}{\partial(x_1, \ldots, x_n)} \triangleq \begin{vmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_m} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_m} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_m} \end{vmatrix} . \tag{3.1}$$

□

Definizione 3.7. *Insieme convesso*
L'insieme $I \subseteq \mathbb{R}^n$ si dice ***convesso*** *se $\forall \mathbf{x}, \mathbf{y} \in I$ e $\forall \beta \in [0, 1]$ si ha che:*

$$\mathbf{z} = \beta\mathbf{x} + (1 - \beta)\mathbf{y} \in I .$$

□

Definizione 3.8. *Funzione convessa*
Una funzione $f : I \to \mathbb{R}$, definita su un insieme convesso $I \subseteq \mathbb{R}^n$ si dice ***convessa*** *se $\forall \mathbf{x}, \mathbf{y} \in I$ e $\forall \beta \in [0, 1]$ si ha che:*

$$f(\beta\mathbf{x} + (1 - \beta)\mathbf{y}) \leq \beta f(\mathbf{x}) + (1 - \beta) f(\mathbf{y}) .$$

□

3.2 Metodo del Gradiente

Il *metodo del gradiente* è un algoritmo per la ricerca dei massimi e minimi di una funzione f, almeno di classe C^1, di due o più variabili. Tale metodo si basa sullo studio del gradiente di f in una successione di punti del dominio di questa.

Ricordiamo che se f è una funzione di classe $C^1(I)$, con I insieme aperto $I \subseteq \mathbb{R}^n$, allora ∇f è il vettore che ha per elementi le derivate parziali di f secondo le componenti di questa. Il gradiente ∇f permette di individuare la direzione di massima variazione della funzione f in ogni suo punto (vedi [20]); mentre nei punti di massimo e di minimo della funzione o, più in generale, nei punti critici, si ha che ∇f si annulla.

Per un problema di ricerca del minimo di una funzione è dunque possibile procedere in questo modo:

1. scegliere un punto di partenza $\mathbf{x}_0 \in I$;
2. calcolare $\nabla f(\mathbf{x}_0)$;
3. muoversi lungo la direzione opposta, o direzione di *antigradiente* di un opportuno passo α_0;
4. se nel nuovo punto il gradiente non si annulla, iterare il procedimento dei punti 1–3 partendo da questo.

Infatti, scegliendo:

$$\mathbf{x}_1 = \mathbf{x}_0 - \alpha_0 \nabla f(\mathbf{x}_0) \ ,$$

si ha che $f(\mathbf{x}_1) \leq f(\mathbf{x}_0)$. Iterando i punti 1–4 appena descritti si ottiene una successione di punti $\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_n$ che converge ad un punto di minimo per f.

Il valore $\alpha_i > 0$ è uno scalare che rappresenta il passo con cui si effettuano gli spostamenti ad ogni iterazione dell'algoritmo. Tale passo può essere mantenuto costante o ricalcolato ad ogni iterazione con tecniche opportune (vedi [20]).

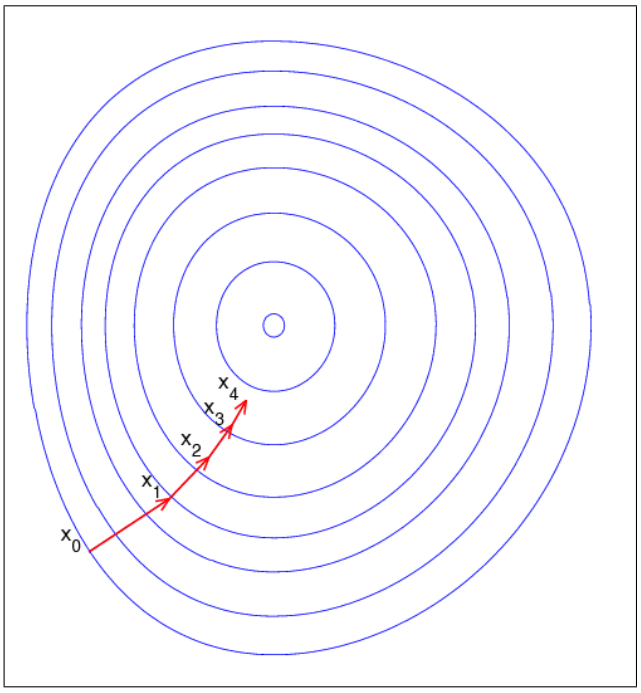


Figura 3.1: Esempio di funzionamento del metodo del gradiente, in rosso la successione di punti $\mathbf{x}_n$, in blu le curve di livello di una funzione di due variabili.

Per problemi di ricerca del massimo il metodo è analogo, ma richiede di spostarsi lungo la direzione del gradiente invece che lungo quella dell'antigradiente.

Si noti che il metodo del gradiente non garantisce di trovare un punto di massimo o minimo assoluto per f, ma solamente un punto critico di f, per cui il gradiente si annulla. Tale problema non si pone nel caso della ricerca del minimo di una funzione convessa, in quanto vale il seguente teorema:

Teorema 2. *Minimo di una funzione convessa*
Sia $f : I \to \mathbb{R}$ una funzione convessa e sia $I \subseteq \mathbb{R}^n$ un insieme convesso. Si ha che se un punto è di minimo relativo per f, allora questo è anche di minimo assoluto per f.

Dimostrazione:
Sia $\mathbf{x}^\star$ un qualunque punto di minimo relativo. Per definizione esiste un intorno $B(\mathbf{x}^\star)$ tale che $f(\mathbf{x}^\star) \leq f(\mathbf{z})$ per ogni $\mathbf{z} \in B(\mathbf{x}^\star)$. Si vuole dimostrare che $f(\mathbf{x}^\star) \leq f(\mathbf{y})$ per ogni $\mathbf{y} \in I$.

Dato un qualunque $\mathbf{y} \in I$ si consideri il punto $\mathbf{z}$ appartenente al segmento che unisce $\mathbf{x}^\star$ ad $\mathbf{y}$ e definito come:

$$z = \beta \mathbf{x}^\star + (1 - \beta)\mathbf{y} \ ,$$

dove $\beta \in [0, 1]$ viene scelto molto vicino al valore 1, in modo che $\mathbf{z} \in B(\mathbf{x}^\star)$. Con questo accorgimento e grazie all'ipotesi di convessità di f si può scrivere:

$$f(\mathbf{x}^\star) \leq f(\mathbf{z}) = f(\beta \mathbf{x}^\star + (1 - \beta)\mathbf{y}) \leq \beta f(\mathbf{x}^\star) + (1 - \beta) f(\mathbf{y}) \ ,$$

da cui dividendo per $(1 - \beta) > 0$ si ottiene:

$$f(\mathbf{x}^\star) \leq f(\mathbf{y}) \ .$$

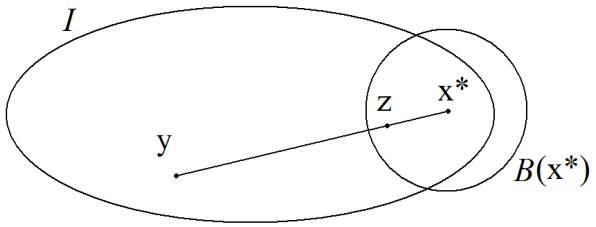


Figura 3.2: Minimo di una funzione convessa

□

3.3 Metodo dei moltiplicatori di Lagrange

Il *metodo dei moltiplicatori di Lagrange* permette di trovare massimi e minimi di una funzione di due o più variabili soggetta a uno o più vincoli, esprimibili in forma funzionale. Il metodo si basa su teoremi matematici, che verranno ora riportati, a titolo esemplificativo, nel caso di una funzione di due variabili soggetta ad un solo vincolo. I concetti verranno poi estesi al caso di funzioni di N variabili soggette a M vincoli.

Sia I un insieme aperto di $\mathbb{R}^2$; siano f e h funzioni di classe $C^1(I)$ e sia L l'insieme dei punti che soddisfano il *vincolo* $h(I) = 0$, ossia:

$$L = \{(x, y) : h(x, y) = 0\} \ .$$

L'obiettivo è trovare i massimi e i minimi della funzione f ristretta all'insieme L, ossia gli *estremi vincolati* di f ad h.

Teorema 3. *Teorema*
Siano I, f e h definiti come sopra; condizione necessaria perché il punto $P =$

(x_1, y_1) di X sia un punto di massimo o minimo relativo di f sotto la condizione $h(x, y) = 0$ è che siano soddisfatte le seguenti equazioni:

$$h(P) = 0 \quad , \qquad \frac{\partial(f,h)}{\partial(x,y)}(P) = 0 \; , \tag{3.2}$$

cioè che il punto P soddisfi il vincolo, e che il determinante della matrice jacobiana associata a f e h calcolato in P sia nullo.

Dimostrazione:
Se $P_0 = (x_0, y_0)$ appartiene all'insieme $L = \{(x, y) : h(x, y) = 0\}$ ed è un punto critico per h, le (3.2) sono verificate.

Se P non è critico per h, allora esiste un intorno T di P, tale che $T \cap L$ sia il grafico di una funzione del tipo $y = g(x)$, funzione definita implicitamente da $h(x, y) = 0$ (vedi Teorema di Dini in [21]). La f ristretta a $T \cap L$ si può quindi rappresentare come $z(x) = f(x, g(x))$. Se P_0 è un punto di massimo o minimo per f ristretta a $T \cap L$ allora deve essere (vedi Derivata di funzioni composte in [21]):

$$\frac{dz(x_0)}{dx} = \frac{\partial f(P_0)}{\partial x} + \frac{\partial f(P_0)}{\partial y} g'(x_0) = 0 \; . \tag{3.3}$$

Ora, ricordando che $g(x)$ è una funzione definita implicitamente da $h(x, y) = 0$ in $T \cap L$, la sua derivata è data da (vedi Derivabilità delle funzioni implicite in [21]):

$$g'(P_0) = -\frac{\frac{\partial h(P_0)}{\partial x}}{\frac{\partial h(P_0)}{\partial y}} \; . \tag{3.4}$$

Usando la (3.3) e la (3.4) si ottiene:

$$\frac{\partial f(P_0)}{\partial x}\frac{\partial h(P_0)}{\partial y} - \frac{\partial f(P_0)}{\partial y}\frac{\partial h(P_0)}{\partial x} = 0 \; ,$$

ovvero:

$$\frac{\partial(f,h)}{\partial(x,y)}(P_0) = 0 \; . \tag{3.5}$$

□

In termini geometrici questo teorema afferma che se P_0 è un punto di estremo vincolato per la f e non è punto critico nè per la f nè per la h, allora la curva di equazione $h(x, y) = 0$ e la cuva di livello passante per $f(P_0)$ hanno in P_0 la stessa tangente. Infatti le equazioni che descrivono le rette tangenti alle due curve sono:

$$\frac{\partial h(P_0)}{\partial x}(x - x_0) + \frac{\partial h(P_0)}{\partial y}(y - y_0) = 0 \; ,$$

$$\frac{\partial f(P_0)}{\partial x}(x - x_0) + \frac{\partial f(P_0)}{\partial y}(y - y_0) = 0 \; ,$$

e la condizione di parallelismo è proprio la (3.5) (vedi [21]).

Il metodo dei moltiplicatori di Lagrange si basa sul risultato del Teorema 3 e fornisce ancora delle condizioni necessarie, ma non sufficienti per la ricerca degli estremi vincolati, nel caso $\nabla h \neq 0$.

Teorema 4. *Teorema dei moltiplicatori di Lagrange*
Siano I un insieme aperto di $\mathbb{R}^2$, f e h due funzioni di classe $C^1(I)$ e si abbia $\nabla h \neq 0$. Si consideri la funzione $\mathcal{L}$, nota come Lagrangiana, delle tre variabili x, y, λ, con $(x, y) \in I$ e $\lambda \in \mathbb{R}$, così definita:

$$\mathcal{L}(x, y, \lambda) \triangleq f(x, y) + \lambda h(x, y) \ , \tag{3.6}$$

e si considerino poi i punti critici della funzione $\mathcal{L}$, cioè le soluzioni (x, y, λ) del sistema:

$$\begin{cases} \dfrac{\partial \mathcal{L}}{\partial x} = 0 \\ \dfrac{\partial \mathcal{L}}{\partial y} = 0 \\ \dfrac{\partial \mathcal{L}}{\partial \lambda} = 0 \end{cases} = \begin{cases} \dfrac{\partial f}{\partial x} + \lambda \dfrac{\partial h}{\partial x} = 0 \\ \dfrac{\partial f}{\partial x} + \lambda \dfrac{\partial h}{\partial x} = 0 \\ h(x, y) = 0 \end{cases} \tag{3.7}$$

I punti (x, y) di massimo o minimo relativo della f, vincolati alla condizione $h(x, y) = 0$, soddisfano al sistema di equazioni (3.7).

Dimostrazione:
le prime due equazioni del sistema (3.7) impongono che i vettori ∇f e ∇h siano linearmente dipendenti, o in termini geometrici, paralleli. Queste sono equivalenti alla seconda condizione (vedi equazione (3.5)) del teorema precedente. Un esempio grafico di come ∇f e ∇h giacciano lungo la stessa direzione in un punto di massimo vincolato è riportato in Figura 2. La terza equazione in (3.7) non

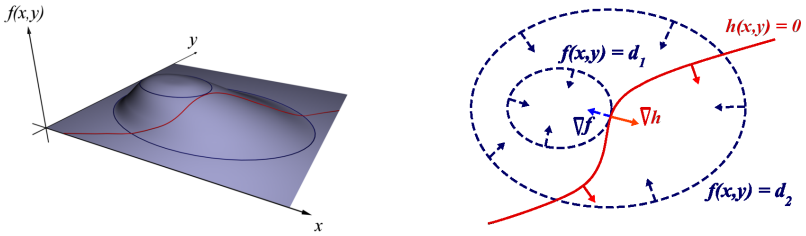


Figura 3.3: Esempio grafico della condizione di parallelismo dei vettori gradiente nel punto di massimo.

è altro che il vincolo stesso, imposto come condizione da soddisfare anche dal teorema precedente. Pertanto le (7) sono conseguenza diretta del teorema 3.

□

Il seguente esempio mostra come si deve procedere per risolvere un problema di ricerca di estremi vincolati utilizzando il metodo dei moltiplicatori di Lagrange.

Esempio 3.1

Trovare i punti di massimo, i punti di minimo, il valore massimo e il valore minimo della funzione $f(x,y) = xy$ sulla circonferenza $S1$ di equazione $x^2 + y^2 - 1 = 0$.

Soluzione:
Per il teorema di Weierstrass sicuramente esistono almeno un punto di massimo e almeno un punto di minimo, perché $f(x,y) = xy$ è continua e $S1$ è un compatto. Si applichi il metodo dei moltiplicatori di Lagrange. Si noti che il gradiente di $x^2 + y^2 - 1$ è:

$$(2x, 2y) \ ,$$

sempre diverso da zero in $S1$. Si trovino i punti stazionari della lagrangiana

$$\mathcal{L}(x,y,\lambda) = xy + \lambda(x^2 + y^2 - 1) \ ,$$

ossia i punti (x,y,λ) nei quali il gradiente di $\mathcal{L}$ si annulla. Il sistema da risolvere è:

$$\begin{cases} y + 2\lambda x = 0 \\ x + 2\lambda y = 0 \\ x^2 + y^2 - 1 = 0 \end{cases} .$$

Dalle prime due equazioni, si ricava

$$-2\lambda = \frac{y}{x} = \frac{x}{y} \ .$$

Si noti che deve essere $x \neq 0$. Infatti, $x = 0$ implica $y = 0$, e quindi la terza equazione del sistema non può essere soddisfatta. Analogamente, si deve avere $y \neq 0$. Allora $x^2 = y^2$. Sostituendo nell'equazione del vincolo $x^2 + y^2 - 1 = 0$ si ottengono i quattro punti:

$$A = \left(\tfrac{1}{\sqrt{2}}; \tfrac{1}{\sqrt{2}}\right) \ , \quad A' = \left(-\tfrac{1}{\sqrt{2}}; -\tfrac{1}{\sqrt{2}}\right) \ ,$$
$$B = \left(-\tfrac{1}{\sqrt{2}}; \tfrac{1}{\sqrt{2}}\right) \ , \quad B' = \left(\tfrac{1}{\sqrt{2}}; -\tfrac{1}{\sqrt{2}}\right) \ .$$

In questo caso, per risolvere il sistema, non è necessario trovare esplicitamente i valori di λ. Si ha $f(A) = f(A') = \frac{1}{2}$ e $f(B) = f(B') = -\frac{1}{2}$. Quindi A e A' sono punti di massimo e B, B' sono punti di minimo. Il valore massimo è $\frac{1}{2}$ e il valore minimo è $-\frac{1}{2}$.

■

Il passaggio da due a più variabili e da uno a più vincoli non presenta difficoltà concettuali, ma solo difficoltà di notazione.

Teorema 5. *Teorema dei moltiplicatori di Lagrange*
Siano $f(x_1, \ldots, x_{k+m})$ e $h_i(x_1, \ldots, x_{k+m})$ con $i = 1, \ldots, m$, funzioni di classe $C^1(\mathbb{R}^{k+m})$ e non sia nullo almeno uno dei $\binom{k+m}{m}$ determinanti jacobiani[2]:

$$\frac{\partial(h_1, \ldots, h_m)}{\partial(x_{i_1}, \ldots, x_{i_m})} \quad , \qquad (1 \leq i_1 \leq \cdots \leq i_m \leq k+m) \ .$$

[2]per la notazione si faccia riferimento all'Eq.(3.1).

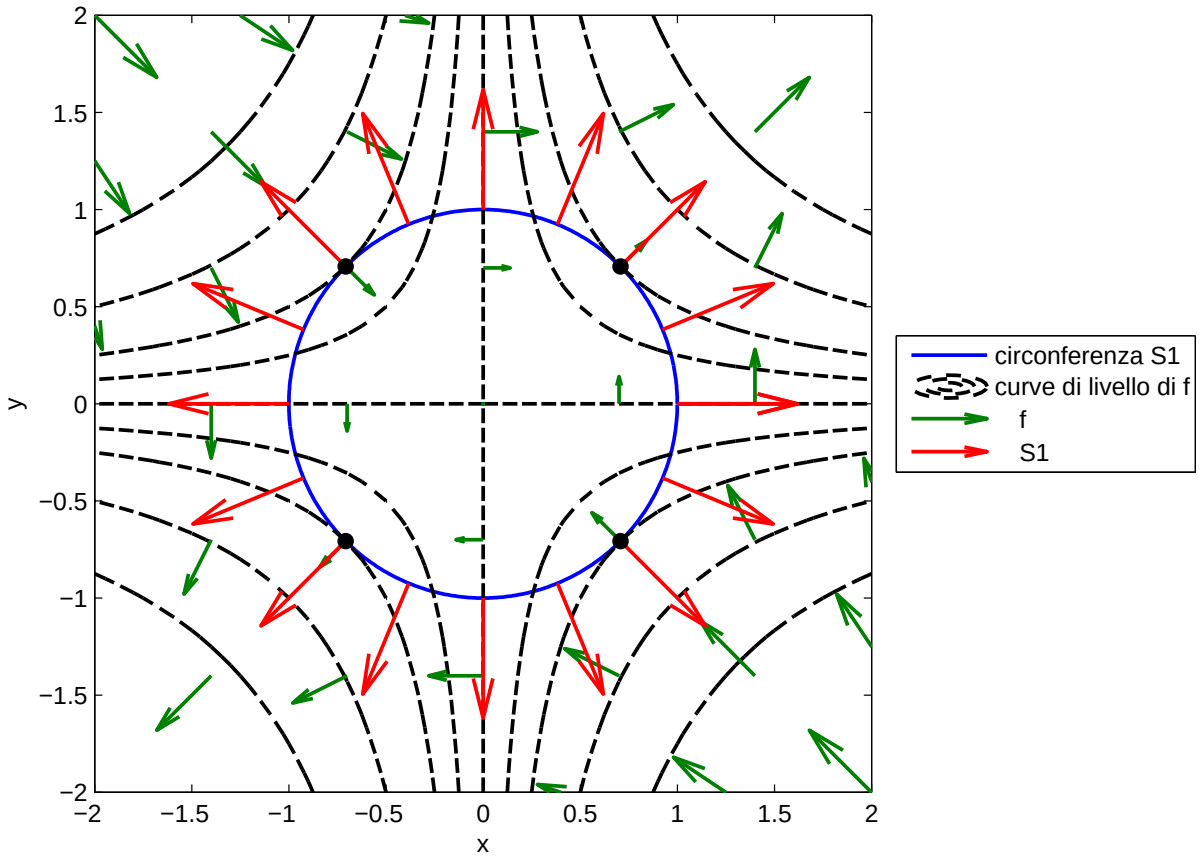


Figura 3.4: Situazione dell'esempio 3.1, si noti il parallelismo dei gradienti nei punti discussi.

Condizione necessaria perché il punto P sia un punto di massimo o minimo per la f sotto le m condizioni:

$$\begin{cases} h_1(x_1, \ldots, x_{k+m}) &= 0 \\ \vdots & \vdots \quad \vdots \\ h_m(x_1, \ldots, x_{k+m}) &= 0 \end{cases} . \tag{3.8}$$

è che esistano i numeri $\lambda_1, \ldots, \lambda_m$ tali che il punto $(P, \lambda_1, \ldots, \lambda_m)$ di $\mathbb{R}^{k+2m}$ sia punto critico per la funzione $\mathcal{L}$ delle $k+2m$ variabili $x_1, \ldots, x_{k+m}, \lambda_1, \ldots, \lambda_m$ così definita:

$$\mathcal{L}(x_1, \ldots, x_{k+m}, \lambda_1, \ldots, \lambda_m) = f + \sum_{i=1}^{m} \lambda_i h_i \, .$$

Il teorema 5 afferma che se un punto $\mathbf{x}_0$ appartenente al dominio vincolato è critico per la funzione f, allora il gradiente di f calcolato in $\mathbf{x}_0$ può essere espresso come combinazione lineare dei gradienti dei vincoli calcolati in $\mathbf{x}_0$.

3.4 Condizioni di Karush – Kuhn – Tucker

Le *condizioni di Karush Kuhn Tucker* (o condizioni KKT) sono condizioni necessarie per la soluzione di un problema di ricerca di estremi vincolati in cui i vincoli soddisfino una condizione di regolarità, detta condizione di qualificazione dei vincoli. Le KKT permettono una generalizzazione del metodo dei moltiplicatori di Lagrange, rendendo possibile la sua applicazione anche a problemi in cui siano presenti vincoli di disuguaglianza.

Per ricavare queste condizioni si prenda in considerazione il seguente problema di ricerca del minimo di una funzione:

$$\begin{aligned} &\min f(\mathbf{x}) \ , \\ &h_i(\mathbf{x}) = 0 \ , \\ &g_j(\mathbf{x}) \leq 0 \ , \end{aligned}$$

in cui $\mathbf{x} \in I \subseteq \mathbb{R}^n$, $f(\mathbf{x})$ di classe $C^1(I)$ è la funzione da minimizzare, $h_i(\mathbf{x})$ con $i = 1, \ldots, K$ sono i vincoli di uguaglianza e $g_j(\mathbf{x})$ con $j = 1, \ldots, M$ sono i vincoli di disuguaglianza.

Lo studio di tale problema permetterà nelle prossime sezioni di introdurre le condizioni KKT dapprima nel caso più semplice, quello in cui sono presenti solo vincoli di uguaglianza, successivamente si considererà un solo vincolo di disuguaglianza e infine si tratterà il caso generale con N equazioni e M disequazioni come vincoli.

Definizione 3.9. *Vincolo attivo in un punto*
Dato un punto $\mathbf{x}_0 \in \mathbb{R}^n$, un vincolo di disuguaglianza $g(\mathbf{x}) \leq 0$ si dice ***attivo nel punto*** $\mathbf{x}_0$ *se la disequazione è soddisfatta all'uguaglianza, ossia:*

$$g(\mathbf{x}_0) = 0 \ .$$

□

Si ricordi inoltre che il gradiente di una funzione $f(\mathbf{x})$ in un punto $\mathbf{x}_0$ è sempre ortogonale alla curva di livello passante per quel punto, ossia al luogo dei punti $f(\mathbf{x}) = f(\mathbf{x}_0)$. Dunque, calcolando il gradiente di un vincolo $h_i(\mathbf{x}_0) = 0$ in un punto $\mathbf{x}_0$, la direzione di $\nabla h_i(\mathbf{x}_0)$ è ortogonale al vincolo.

3.4.1 Condizioni KKT per soli vincoli di uguaglianza

Come si è già visto con il metodo dei moltiplicatori di Lagrange (vedi Teorema 4), quando si considerano solo vincoli di uguaglianza il problema di minimizzazione vincolata si può ridurre a un problema di ricerca dei punti critici della funzione Lagrangiana:

$$\mathcal{L}(\mathbf{x}, \lambda) = f(\mathbf{x}) + \lambda_1 h_1(\mathbf{x}) + \cdots + \lambda_K h_K(\mathbf{x}) \ .$$

Ossia, il minimo di f è tra quei punti che soddisfano il sistema:

$$\begin{cases} \nabla_x \mathcal{L}(\mathbf{x}, \lambda) = \nabla f(\mathbf{x}) + \lambda_1 \nabla h_1(\mathbf{x}) + \cdots + \lambda_K \nabla h_K(\mathbf{x}) = 0 \\ \nabla_{\lambda_i} \mathcal{L}(\mathbf{x}, \lambda) = h_i(\mathbf{x}) = 0 \qquad\qquad i = 1, \ldots, K \end{cases} , \tag{3.9}$$

o, equivalentemente:

$$\begin{cases} \nabla f(\mathbf{x}) = \lambda_1 \nabla h_1(\mathbf{x}) + \cdots + \lambda_N \nabla h_N(\mathbf{x}) \\ h_i(\mathbf{x}) = 0 \qquad\qquad i = 1, \ldots, K \end{cases} . \tag{3.10}$$

L'equazione (3.10) dice che un punto $\mathbf{x}_0$ è critico per la funzione f, ristretta al dominio vincolato, se il gradiente di f calcolato in $\mathbf{x}_0$ può essere espresso come combinazione lineare dei gradienti dei vincoli calcolati in $\mathbf{x}_0$. Si noti che i coefficienti di questa combinazione lineare corrispondono ai moltiplicatori di Lagrange $(\lambda_1, \ldots, \lambda_K)$. Come si è visto in sezione 3.3, nel caso di un solo vincolo questo comporta che il gradiente di f e il gradiente del vincolo siano paralleli.

Il seguente esempio conferma i risultati appena enunciati.

Esempio 3.2

Si supponga di voler calcolare il minimo della funzione

$$f(x_1, x_2) = x_1 + x_2 \ ,$$

sui punti che rispettano la condizione

$$h_1(x_1, x_2) = x_1^2 + x_2^2 - 2 = 0 \ ,$$

cioè sulla circonferenza di raggio $\sqrt{2}$ e centro nell'origine (Figura 3).

Il punto di minimo è $\mathbf{x}^\star = (-1, -1)$, e di conseguenza il minimo vale -2. Da ogni altro punto della circonferenza infatti, è possibile muoversi in modo da mantenere l'ammissibilità (ossia, di rimanere sulla circonferenza) e contemporaneamente diminuire la funzione obiettivo. Ad esempio, dal punto (-1, 1), la funzione migliora spostandosi lungo la circonferenza in senso antiorario. Si osservi che il gradiente della funzione obiettivo è $\nabla f(\mathbf{x}) = [1, 1]$, indipendente da $\mathbf{x} \in \mathbb{R}^2$, mentre $\nabla h_1(\mathbf{x}) = [2x_1, 2x_2]$. Osservando la Fig.3, si può vedere che nel punto $\mathbf{x}^\star$, il

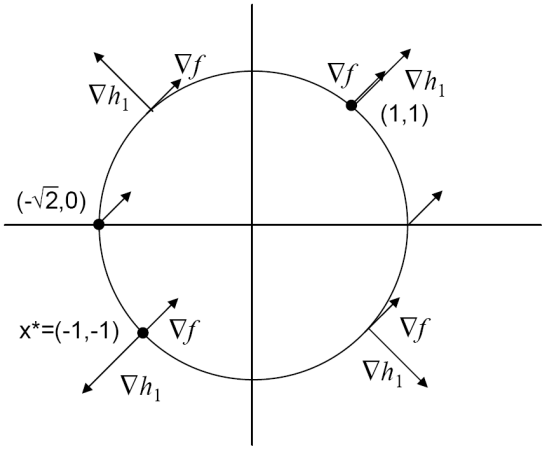


Figura 3.5: Rappresentazione grafica della situazione dell'esempio 3.2

gradiente $\nabla f(\mathbf{x}^\star)$ e la normale al vincolo $\nabla h_1(\mathbf{x}^\star)$ sono paralleli, e

$$\nabla f(\mathbf{x}^\star) = -\frac{1}{2} \nabla h_1(\mathbf{x}^\star) \ ,$$

ossia vale la ((3.10)) con $\lambda_1 = -\frac{1}{2}$. Come si vede, anche nel punto (1,1) i due vettori sono paralleli. Questo infatti è un altro punto estremo vincolato della funzione f, più precisamente, (1,1) è il punto di massimo assoluto della f calcolata nei punti appartenenti alla circonferenza h_1.

■

Un'importante osservazione è che il segno dei moltiplicatori λ_i non ha particolare significato nel caso i vincoli siano espressi unicamente da equazioni. Infatti, nella formulazione del problema, si sarebbe potuto rappresentare ogni vincolo, anziché con $h_i(\mathbf{x}) = 0$ con $-h_i(\mathbf{x}) = 0$. La soluzione del problema ovviamente non sarebbe cambiata, ma per soddisfare la (3.9) si sarebbe dovuto scegliere moltiplicatori di segno opposto. Dunque, se $\mathbf{x}_0$ è un punto di minimo di un problema con un solo vincolo di uguaglianza, $\nabla f(\mathbf{x}_0)$ e $\nabla h_1(\mathbf{x}_0)$ possono avere lo stesso verso o verso opposto, ma l'essenziale è che siano paralleli (cioè abbiano la stessa direzione). Come si vedrà ora, tali considerazioni non sono più valide nel caso in cui siano presenti disequazioni.

3.4.2 Condizioni KKT con una sola disequazione per vincolo

Si consideri ora il caso in cui il dominio vincolato L sia definito da una singola disequazione. Sia $\mathbf{x}_0$ un punto appartenente a L, ossia tale che $g_1(\mathbf{x}_0) \leq 0$, e si supponga $\nabla g_1(\mathbf{x}_0) \neq 0$. Partendo da $\mathbf{x}_0$ si vogliono formulare delle condizioni che, se verificate, portano a escludere che $\mathbf{x}_0$ possa essere punto di minimo e formulare così delle condizioni necessarie affinchè $\mathbf{x}_0$ sia un punto stazionario (e,se la funzione è convessa, di minimo).

Per quanto concerne la funzione f, se $\mathbf{x}_0$ si non è un punto di minimo, esiste un vettore direzione $\mathbf{d}$ tale che:

$$\nabla f(\mathbf{x}_0)^T \mathbf{d} < 0 \, . \tag{3.11}$$

Per quanto riguarda il vincolo g_1, usando la formula di Taylor del primo ordine, si può scrivere (vedi [21]):

$$g_1(\mathbf{x}_0 + \mathbf{d}) \approx g_1(\mathbf{x}_0) + \nabla g_1(\mathbf{x}_0)^T \mathbf{d} \, ,$$

e dunque perchè il punto $\mathbf{x}_0 + \mathbf{d}$ soddisfi il vincolo deve essere:

$$g_1(\mathbf{x}_0) + \nabla g_1(\mathbf{x}_0)^T \mathbf{d} \leq 0 \, . \tag{3.12}$$

Per stabilire ora se esiste una direzione $\mathbf{d}$ tale da soddisfare (3.11) e (3.12), si distingua il caso in cui $\mathbf{x}_0$ sia all'interno dell'insieme $L = \{\mathbf{x} | g_1(\mathbf{x}) \leq 0\}$ da quello in cui giaccia, invece, sulla frontiera.

- **Caso 1.** Se $\mathbf{x}_0$ è interno alla regione ammissibile, cioè $g_1(\mathbf{x}_0) > 0$ o equivalentemente il vincolo g_1 non è attivo, allora è sempre possibile trovare un vettore di spostamento $\mathbf{d}$ che soddisfi la (3.12), a patto di sceglierlo abbastanza piccolo in norma in modo che $\mathbf{x}_0 + \mathbf{d}$ cada ancora nella regione

ammissibile. Dunque, l'unica possibilità perché, a partire dal punto $\mathbf{x}_0$, non esista una direzione di discesa per f (o che, equivalentemente, $\mathbf{x}_0$ sia punto stazionario) è che sia:

$$\nabla f(\mathbf{x}_0) = 0 \ . \tag{3.13}$$

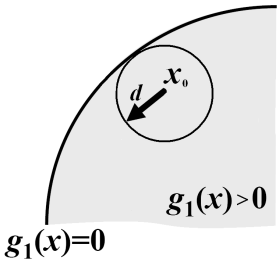


Figura 3.6: Rappresentazione grafica del caso 1

.

- **Caso 2.** Si supponga ora che $\mathbf{x}_0$ appartenga alla frontiera di X, e quindi $g_1(\mathbf{x}_0) = 0$, ossia il vincolo è attivo in $\mathbf{x}_0$, le condizioni (3.11) e (3.12), da soddisfare simultaneamente, diventano:

$$\nabla f(\mathbf{x}_0)^T \mathbf{d} < 0 \ , \tag{3.14}$$

$$\nabla g_1(\mathbf{x}_0)^T \mathbf{d} \leq 0 \ . \tag{3.15}$$

Queste due condizioni definiscono rispettivamente un semispazio aperto e uno chiuso. Se la loro intersezione non è vuota, è possibile individuare una direzione di spostamento che porta da $\mathbf{x}_0$ ad un punto $\mathbf{x}_1 \triangleq \mathbf{x}_0 + \mathbf{d}$ ancora $\in L$ e tale che $f(\mathbf{x}_1) < f(\mathbf{x}_0)$. Ora, l'unico caso in cui non esista una direzione $\mathbf{d}$ che soddisfi simultaneamente la (3.14) e la (3.15) è quello in cui $\nabla f(\mathbf{x}_0)$ e $\nabla g_1(\mathbf{x}_0)$ abbiano verso opposto, cioè esista un $\lambda_1 \geq 0$ tale che:

$$\nabla f(\mathbf{x}_0) = -\lambda_1 \nabla g_1(\mathbf{x}_0) \ . \tag{3.16}$$

Si noti che stavolta il segno del moltiplicatore è importante. Se infatti nella (3.16) λ_1 fosse negativo, $\nabla f(\mathbf{x}_0)$ e $\nabla g_1(\mathbf{x}_0)$ avrebbero lo stesso verso e i due semispazi definiti dalle (3.14) e (3.15) verrebbero a coincidere (a meno della frontiera); in tal modo qualunque $\mathbf{d}$ in tale semispazio aperto soddisferebbe le condizioni (3.11) e (3.12). Si veda a tal proposito la Figura (3.7).

Si consideri ora la funzione lagrangiana:

$$\mathcal{L}(\mathbf{x}, \lambda_1) = f(\mathbf{x}) + \lambda_1 g_1(\mathbf{x}) \ ,$$

il cui studio dei punti stazionari ci permette di unificare i due sotto-casi esaminati (vedi equazioni (3.13) e (3.16)). Si può infatti concludere che se, il punto $\mathbf{x}^\star$, è

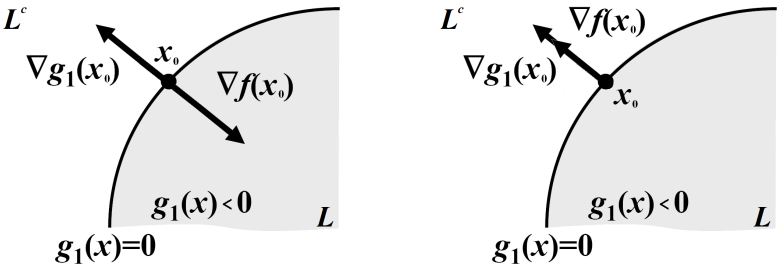


Figura 3.7: caso a) $\lambda_1 \geq 0$ i punti $\mathbf{x}$ tali che $f(\mathbf{x}) < f(\mathbf{x}_0)$ si trovano solo nella regione $L^c \Rightarrow \mathbf{x}_0$ è un punto di minimo vincolato. caso b) $\lambda_1 \leq 0$ i punti $\mathbf{x}$ tali che $f(\mathbf{x}) < f(\mathbf{x}_0)$ si possono trovare all'interno della regione ammissibile $L \Rightarrow \mathbf{x}_0$ non è punto di minimo vincolato.

un punto di minimo per f, allora esiste un moltiplicatore lagrangiano $\lambda_1^\star \geq 0$ tale che risultano soddisfatte le due condizioni:

$$\nabla_x \mathcal{L}(\mathbf{x}^\star, \lambda_1^\star) = 0 \,, \tag{3.17}$$

$$\lambda_1^\star g_1(\mathbf{x}^\star) = 0 \,. \tag{3.18}$$

La (3.18) è nota come condizione di complementarietà; essa è verificata per $\lambda_1^\star = 0$ o per $\lambda_1^\star \neq 0$ e $g_1(\mathbf{x}^\star) = 0$. Se $g_1(\mathbf{x}^\star) = 0$ il vincolo in $\mathbf{x}^\star$ è attivo (caso 2) e la (3.17) coincide con la (3.16). Se invece il vincolo non è attivo (caso 1), per soddisfare la (3.18) deve essere $\lambda_1^\star = 0$ e di conseguenza la (3.17) diventa $\nabla f(\mathbf{x}^\star) = 0$ e cioè la condizione perchè $\mathbf{x}^\star$ sia un punto critico per f. Si noti che può anche accadere che $\lambda_1^\star = 0$ pur essendo il vincolo attivo nel punto $\mathbf{x}^\star$.

Esempio 3.3

Si consideri ancora l'esempio 3.2, e si estenda la regione ammissibile a tutto il cerchio delimitato dalla circonferenza. Il problema diventa

$$f(x_1, x_2) = x_1 + x_2 \,,$$

$$g_1(x_1, x_2) = x_1^2 + x_2^2 - 2 \leq 0 \,.$$

Soluzione:

$\nabla f = (1, 1)$ non si annnulla mai, quindi non ci sono punti di minimo tra i punti interni del cerchio. Come nell'esempio 3.2, il vettore $\nabla g_1(\mathbf{x})$ calcolato in un qualsiasi punto $\mathbf{x}$ sulla sulla circonferenza, punta verso l'esterno di questa. Si osservi che il minimo della funzione f si trova ancora in corrispondenza del punto $\mathbf{x}^\star = (-1, -1)$, e che vale la condizione:

$$\nabla f(\mathbf{x}^\star) = -\lambda_1 \nabla g_1(\mathbf{x}^\star) \,,$$

con $\lambda_1 = 1/2$ (si noti che $\lambda_1 \geq 0$). Nel punto di massimo (1,1) vale ancora la condizione, ma con moltiplicatore negativo.

■

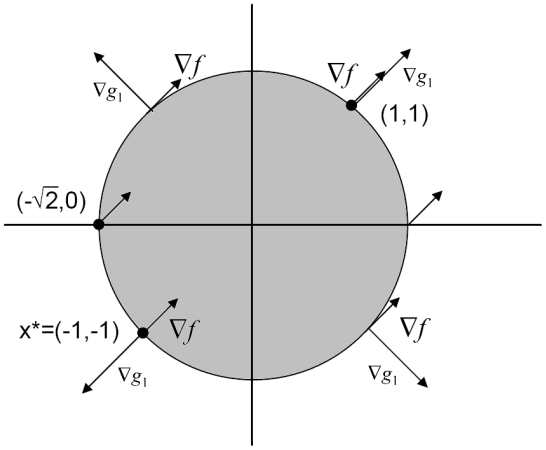


Figura 3.8: Rappresenzazione grafica della situazione dell'esempio 3.3

3.4.3 Condizioni KKT con più disequazioni per vincolo

Vengono riportate alcune definizioni utili.

Definizione 3.10. *Combinazione conica*
Dato un insieme di punti $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k \in \mathbb{R}^n$*, e* k *scalari* $\lambda_1 \geq 0, \lambda_2 \geq 0, \ldots, \lambda_k \geq 0$ *non tutti nulli, si dice* ***combinazione conica*** *dei punti* $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k$ *la quantità:*

$$\lambda_1 \mathbf{x}_1 + \lambda_2 \mathbf{x}_2 + \cdots + \lambda_k \mathbf{x}_k \ .$$

Si dice cono generato da $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k$ *l'insieme di tutte le possibili combinazioni coniche di* $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k$*.*

Definizione 3.11. *Insieme dei vincoli attivi in un punto*
Dato un insieme $L = \{\mathbf{x} \in \mathbb{R}^n | h_i(\mathbf{x}) = 0$ *e* $g_j(\mathbf{x}) \geq 0, i = 1, \ldots, K, j = 1, \ldots, M\}$ *e un punto* $\mathbf{x}_0 \in L$*, si dice* ***insieme dei vincoli attivi in*** $\mathbf{x}_0$ *l'insieme degli indici* j *tali per cui* $g_j(\mathbf{x}_0) = 0$*, in formule:*

$$I_a(\mathbf{x}_0) = \{j | g_j(\mathbf{x}_0) = 0, j = 1, \ldots, M\} \ .$$

□

Definizione 3.12. *Punto di regolarità*
Dato un insieme $L = \{\mathbf{x} \in \mathbb{R}^n : h_i(\mathbf{x}) = 0, g_j(\mathbf{x}) \geq 0\}$*, un punto* $\mathbf{x}_0 \in L$*, e il corrispondente insieme di vincoli attivi* $I_a(\mathbf{x}_0)$*, si dice che i vincoli attivi soddisfano la condizione di qualificazione in* $\mathbf{x}_0$ *se i gradienti* $\nabla h_i(\mathbf{x}_0)$ *con* $i = 1, \ldots, K$ *e* $\nabla g_j(\mathbf{x}_0)$ *con* $j \in I_a(\mathbf{x}_0)$*, sono tra loro linearmente indipendenti. Un punto* $\mathbf{x}_0$ *per il quale valga la condizione di qualificazione dei vincoli si dice* ***punto di regolarità****.*

Si noti che la condizione di qualificazione dei vincoli attivi in $\mathbf{x}_0$ equivale a richiedere che siano linearmente indipendenti le righe della matrice jacobiana relativa ai vincoli $h_i(\mathbf{x}_0) \quad \forall i$ e $g_j(\mathbf{x}_0) \quad \forall j \in I_a(\mathbf{x}_0)$, ossia che:

$$\frac{\partial(h_1, \ldots, h_K, g_{j_1}, \ldots, g_{j_l})}{\partial(x_1, \ldots, x_n)} = 0 \quad , \quad j_1, \ldots, j_l \in I_a(\mathbf{x}_0) \ .$$

Si supponga di avere due vincoli di disuguaglianza $g_1(\mathbf{x}) \leq 0$ e $g_2(\mathbf{x}) \leq 0$ (il ragionamento per il caso di m vincoli è il medesimo). Si consideri un punto $\mathbf{x}_0 \in L$, si vogliono definire le condizioni per le quali $\mathbf{x}_0$ risulta un punto stazionario. Nel caso in cui in $\mathbf{x}_0$ solo uno o nessuno dei due vincoli risulti attivo, ci si riconduce ad uno dei casi discussi precedentemente. La situazione è invece diversa nel caso in cui, in $\mathbf{x}_0$, ambedue i vincoli siano attivi. Si supponga che $\mathbf{x}_0$ sia un punto di regolarità. Il vettore $\mathbf{d}$ è una "direzione di discesa ammissibile" se valgono:

$$\nabla g_1(\mathbf{x}_0)^T \mathbf{d} \leq 0 \ , \tag{3.19}$$

$$\nabla g_2(\mathbf{x}_0)^T \mathbf{d} \leq 0 \ , \tag{3.20}$$

$$\nabla f(\mathbf{x}_0)^T \mathbf{d} < 0 \ . \tag{3.21}$$

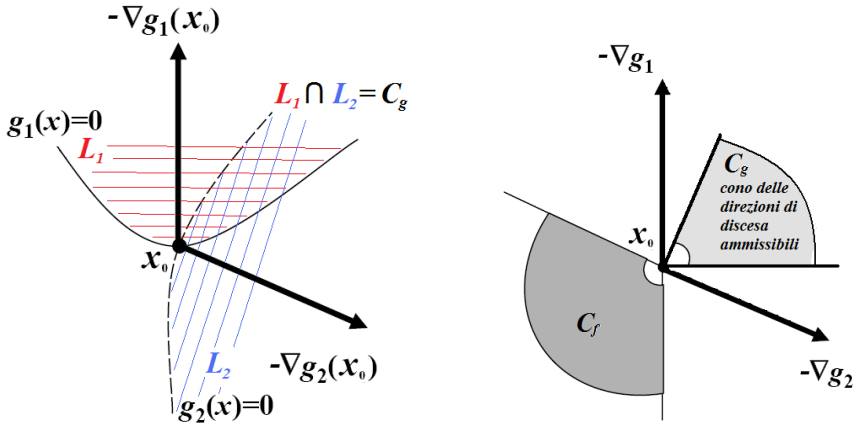


Figura 3.9: (a) i due vincoli g_1 e g_2 con le rispettive regioni ammissibili; (b) il cono delle direzioni ammissibili C_g e la regione C_f in cui deve trovarsi $-\nabla f$ affinchè $\mathbf{x}_0$ sia punto di minimo vincolato.

Come mostrato in Figura 3.9, gli antigradienti dei vincoli individuano un cono C_g di direzioni ammissibili che è tanto più stretto quanto più grande è l'angolo tra questi. Ora, se $\mathbf{x}_0$ è un punto di minimo vincolato per f, nessuna delle direzioni contenute in C_g può essere di discesa per f; le direzioni di discesa per f (e dunque l'antigradiente di f) devono essere contenute in un cono C_f tale che $C_f \cap C_g = \emptyset$, come illustrato in Figura 3.9b. Di conseguenza, osservando la figura, si vede che, se un punto $\mathbf{x}_0$ è un punto di minimo, allora il gradiente $\nabla f(\mathbf{x}_0)$ deve essere contenuto nel cono generato da $-\nabla g_1(\mathbf{x}_0)$ e $-\nabla g_2(\mathbf{x}_0)$.

Volendo esprimere anche in questo caso le condizioni necessarie in una forma analoga a (3.17) e (3.18), si scriva la funzione Lagrangiana:

$$\mathcal{L}(\mathbf{x}, \lambda) = f(\mathbf{x}) + \lambda_1 g_1(\mathbf{x}) + \lambda_2 g_2(\mathbf{x}) \ ,$$

dove λ indica il vettore $[\lambda_1, \lambda_2]^T$ dei moltiplicatori lagrangiani. Se $\mathbf{x}^\star$ è un punto di minimo, allora esiste un vettore $\lambda^\star \geq 0$ di moltiplicatori tale che:

$$\nabla_x \mathcal{L}(\mathbf{x}^\star, \lambda^\star) = 0 \ , \tag{3.22}$$

$$\lambda_1^\star g_1(\mathbf{x}^\star) = 0 \ , \tag{3.23}$$

$$\lambda_2^\star g_2(\mathbf{x}^\star) = 0 \ . \tag{3.24}$$

In particolare, la (3.22) indica che

$$\nabla_x \mathcal{L}(\mathbf{x}^\star, \lambda^\star) = \nabla f(\mathbf{x}^\star) + \lambda_1 \nabla g_1(\mathbf{x}^\star) + \lambda_2 \nabla g_2(\mathbf{x}^\star) = 0 \ ,$$

cioè che il gradiente di f è ottenibile come combinazione conica degli antigradienti dei vincoli attivi in $\mathbf{x}^\star$, (i.e. $-\nabla f \in C_f$).

Esempio 3.4

Riprendendo il problema presentato negli esempi 3.2 e 3.3, si introduca ora l'ulteriore vincolo $x_2 \geq 1$. Il problema diventa:

$$f(x_1, x_2) = x_1 + x_2 \ ,$$
$$g_1(x_1, x_2) = x_1^2 + x_2^2 - 2 \leq 0 \ ,$$
$$g_2(x_1, x_2) = 1 - x_2 \leq 0 \ ,$$

e la regione ammissibile è quella mostrata in Figura 3.10.

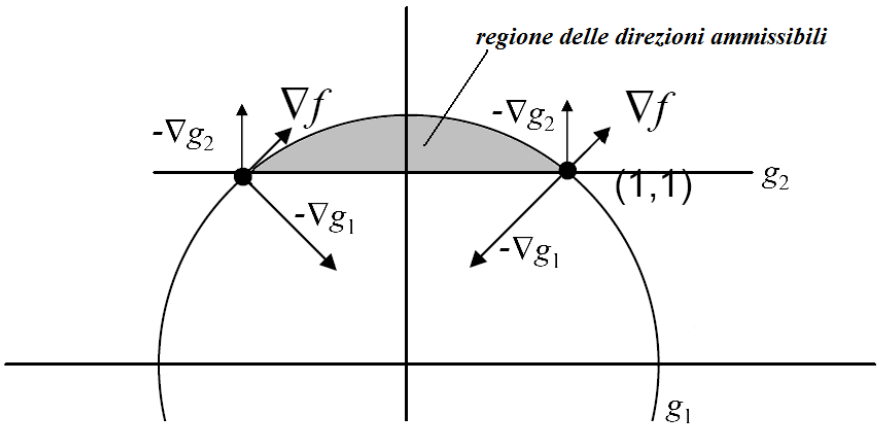


Figura 3.10: Rappresentazione grafica della situazione dell'esempio 3.4

Soluzione:
La soluzione ottima è $\mathbf{x}^\star = (-1, 1)$, punto nel quale ∇f si può esprimere come combinazione conica (eq. 3.22) di $-\nabla g_1$ e $-\nabla g_2$ scegliendo il vettore di coefficienti $\lambda = [\frac{1}{2}, 2]$. Si noti che, in quest caso, le condizioni di complementarietà sono soddisfatte con $\lambda \neq 0$ poichè i vincoli sono attivi in $\mathbf{x}^\star$. Inoltre, in questo esempio, tutti i punti di L sono punti di regolarità, $\forall \mathbf{x} \in L$ con $\mathbf{x} \neq \mathbf{x}^\star$ si può verificare che le condizioni (3.22)–(3.24) non sono soddisfatte. Si prenda ad esempio il punto $\hat{\mathbf{x}} = (1, 1)$. Anche qui, ambedue i vincoli sono attivi. Il gradiente della f non giace però nel cono individuato da $-\nabla g_1$ e $-\nabla g_2$. Quindi, è possibile trovare una direzione di discesa ammissibile, come ad esempio è $\mathbf{d} = [-1, 0]^T$ e $\hat{\mathbf{x}}$ non è punto

di minimo vincolato. $\forall \tilde{\mathbf{x}} \in L \quad \tilde{\mathbf{x}} \neq \hat{\mathbf{x}} \quad \tilde{\mathbf{x}} \neq \mathbf{x}^\star$, invece, le condizioni di complementarietà implicano $\lambda = 0$ da cui, perchè $\tilde{\mathbf{x}}$ sia punto stazionario, $\nabla f(\tilde{\mathbf{x}}) = 0$, mai verificato.

■

In caso di M vincoli del tipo $g_j \leq 0$ e punti di regolarità le (3.22)–(3.24) si generalizzano come descritto nella prossima sezione. Nei punti di non regolarità invece il problema deve essere trattato caso per caso (vedi [22]).

3.4.4 Condizioni di Karush – Kuhn – Tucker

Definizione 3.13. *Funzione Lagrangiana*
Dato un problema di ricerca dei minimi vincolati di una funzione f, e i relativi vincoli:

$$h_i(\mathbf{x}) = 0 \ ,$$

$$g_j(\mathbf{x}) \geq 0 \ ,$$

con $i \in \mathcal{E} = \{1, \ldots, K\}$, $j \in \mathcal{I} = \{1, \ldots, M\}$ e $\mathbf{x} \in \mathbb{R}^n$ si definisce ***funzione lagrangiana*** *la funzione $\mathbb{R}^{n+|\mathcal{E}|+|\mathcal{I}|} \to \mathbb{R}$:*

$$\mathcal{L}(\mathbf{x}, \lambda) = f(\mathbf{x}) + \sum_{i \in \mathcal{E}} \lambda_i h_i(\mathbf{x}) + \sum_{j \in \mathcal{I}} \lambda_j g_j(\mathbf{x}) \ . \tag{3.25}$$

Teorema 6. *Condizioni KKT*
Sia $\mathbf{x}^\star$ sia un minimo locale di un problema di minimizzazione vincolata, e sia $\mathbf{x}^\star$ un punto di regolarità. Allora esiste un vettore $\lambda^\star$, avente componenti $\lambda_k^\star$ con $k \in \mathcal{E} \cup \mathcal{I}$ tale che:

$$\nabla_x \mathcal{L}(\mathbf{x}^\star, \lambda^\star) = 0 \ , \tag{3.26}$$

$$h_i(\mathbf{x}^\star) = 0 \qquad \forall i \in \mathcal{E} \ , \tag{3.27}$$

$$g_j(\mathbf{x}^\star) \geq 0 \qquad \forall j \in \mathcal{I} \ , \tag{3.28}$$

$$\lambda_k^\star \geq 0 \qquad \forall k \in \mathcal{E} \cup \mathcal{I} \ , \tag{3.29}$$

$$\lambda_j^\star g_j(\mathbf{x}^\star) = 0 \qquad \forall j \in \mathcal{I} \ . \tag{3.30}$$

□

La (3.26) esprime la condizione di annullamento del gradiente della funzione lagrangiana associata al problema, o condizione di stazionarietà, le condizioni (3.27) e (3.28) rappresentano i vincoli di ammissibilità del punto $\mathbf{x}^\star$, la (3.29) impone la condizione di non negatività dei moltiplicatori Lagrangiani, e infine la (3.30) è la condizione di complementarietà.

3.5 Algoritmo di Nesterov

3.5.1 Algoritmo di minimizzazione per funzioni regolari

Il metodo proposto da Nesterov in [23, 24] può risolvere problemi di minimizzazione su una qualsiasi funzione f regolare (*smooth*) definita su un insieme convesso $\mathcal{Q}_p$. Una funzione si dice regolare se è derivabile infinite volte, cioè è di classe C^∞.

La classe di problemi in questione è dunque del tipo:

$$\min_{\mathbf{x}\in\mathcal{Q}_p} f(\mathbf{x}) \ . \tag{3.31}$$

L'insieme $\mathcal{Q}_p$ viene chiamato insieme primale possibile. Si assume che la funzione f sia differenziabile, che il suo gradiente $\nabla f(\mathbf{x})$ sia Lipschitziano, ovvero che valga la relazione:

$$\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\|_{\ell_2} \leq L\|\mathbf{x}-\mathbf{y}\|_{\ell_2} \ , \tag{3.32}$$

dove L è un upper bound sulla costante di Lipschitz.

Se sono rispettate le ipotesi appena descritte l'algoritmo di Nesterov minimizza $f(\mathbf{x})$ su $\mathcal{Q}_p$ attraverso il calcolo iterativo di tre successioni $\{\mathbf{x}_k\}$, $\{\mathbf{y}_k\}$ e $\{\mathbf{z}_k\}$. L'algoritmo dipende inoltre da due successioni di scalari $\{\alpha_k\}$ e $\{\tau_k\}$ che saranno scelte in modo da garantire la convergenza. Di seguito vengono riportati i passi del metodo di Nesterov:

Inizializza $\mathbf{x}_0$.

For $k \geq 0$,

1. Calcola $\nabla f(\mathbf{x}_k)$.
2. Calcola $\mathbf{y}_k$:

$$\mathbf{y}_k = \arg\min_{\mathbf{x}\in\mathcal{Q}_p} \frac{L}{2}\|\mathbf{x}-\mathbf{x}_k\|_{\ell_2}^2 + \langle\nabla f(\mathbf{x}_k), \mathbf{x}-\mathbf{x}_k\rangle.$$

3. Calcola $\mathbf{z}_k$:

$$\mathbf{z}_k = \arg\min_{\mathbf{x}\in\mathcal{Q}_p} \frac{L}{\sigma_p} p_p(\mathbf{x}) + \sum_{i=0}^{k} \alpha_i \langle\nabla f(\mathbf{x}_i), \mathbf{x}-\mathbf{x}_i\rangle.$$

4. Aggiorna $\mathbf{x}_k$:

$$\mathbf{x}_k = \tau_k\mathbf{z}_k + (1-\tau_k)\mathbf{y}_k \ .$$

Stop quando è soddifatta una condizione di arresto.

All'iterazione k, $\mathbf{y}_k$ rappresenta la stima sulla soluzione ottima. Applicando l'algoritmo fino al al passo 2, utilizzando $\mathbf{y}_{k-1}$ al posto di $\mathbf{x}_k$, si ottiene un metodo del primo ordine standard con rate di convergenza $\mathcal{O}(1/k)$.

La novità introdotta da Nesterov con il suo algoritmo è la presenza della sequenza $\mathbf{z}_k$ che "memorizza"le iterazioni precedenti attraverso la somma pesata dei gradienti calcolati in precedenza. Un altro aspetto introdotto nel passo 3 è l'uso di una funzione di prossimità (*prox–function*) $p_p(\mathbf{x})$. Questa funzione deve essere fortemente convessa, con parametro di covessità σ_p; inoltre se si impone che la funzione si annnulli in un punto $\mathbf{x}_p^c = \arg\min_{\mathbf{x}} p_p(\mathbf{x})$ si ha che:

$$p_p(\mathbf{x}) \geq \frac{\sigma_p}{2}\|\mathbf{x} - \mathbf{x}_p^c\|_{\ell_2}^2 \ .$$

La funzione di prossimità deve essere scelta in modo che $\mathbf{x}_p^c \in \mathcal{Q}_p$ così da evitare che $\mathbf{z}_k$ si allontani troppo dal centro $\mathbf{x}_p^c$.

L'aggiornamento di $\mathbf{x}_k$ avviene mediante una media pesata tra $\mathbf{y}_k$ e $\mathbf{z}_k$. Nelle prime iterazioni dell'algoritmo viene data la stessa importanza alle due successioni, mentre nel seguito $\mathbf{z}_k$ viene tenuta in considerazione in modo sempre minore, per evitare che una scelta sbagliata del centro $\mathbf{x}_p^c$ influisca in modo negativo sulla soluzione.

Nesterov ha dimostrato in [23] le condizioni sulla scelta delle successioni $\{\alpha_k\}$ e $\{\tau_k\}$ per garantire la convergenza del metodo. Inoltre propone come possibili successioni le seguenti:

$$\alpha_k = \frac{k+1}{2} \ , \tag{3.33}$$

$$\tau_k = \frac{2}{k+3} \ . \tag{3.34}$$

Utilizzando questi valori per α_k e τ_k e procedendo come indicato nei passi dell'algoritmo, la soluzione converge a:

$$\mathbf{x}^\star = \arg\min_{\mathbf{x}\in\mathcal{Q}_p} f(\mathbf{x}) \ , \tag{3.35}$$

e la velocità di convergenza risulta:

$$f(\mathbf{y}_k) - f(\mathbf{x}^\star) \leq \frac{4Lp_p(\mathbf{x}^\star)}{(k+1)^2\sigma_p} \ , \tag{3.36}$$

che risulta migliore di quello che si può ottenere con tecniche standard basate sul gradiente, in quanto si ha che l'approssimazione scala come L/k^2 al posto di L/k.

3.5.2 Approssimazione regolare di funzioni non differenziabili

Recentemente Nesterov ha esteso l'applicabilità del suo algoritmo anche a funzioni convesse non regolari [25]. Assumendo che la funzione f si possa scrivere come:

$$f(\mathbf{x}) = \max_{\mathbf{u}\in\mathcal{Q}_d} \langle \mathbf{u}, \mathbf{W}\mathbf{x}\rangle \ , \tag{3.37}$$

dove $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{u} \in \mathbb{R}^p$ e $\mathbf{W} \in \mathbb{R}^{p \times n}$. L'insieme $\mathcal{Q}_d$ viene chiamato insieme duale possibile, e si suppone che sia convesso.
Tutte queste ipotesi risultano verificate nei casi di interesse di questa tesi, in particolare per la funzione $\|\mathbf{x}\|_{\ell_1}$.

Usando la scrittura (3.37) si può riscrivere il problema (3.31) nel seguente problema del punto di sella:

$$\min_{\mathbf{x} \in \mathcal{Q}_p} f(\mathbf{x}) = \min_{\mathbf{x} \in \mathcal{Q}_p} \max_{\mathbf{u} \in \mathcal{Q}_d} \langle \mathbf{u}, \mathbf{W}\mathbf{x} \rangle \ . \tag{3.38}$$

In generale però f è convessa, ma non regolare. In [25] Nesterov propone come approssimazione regolare la seguente:

$$f_\mu(\mathbf{x}) = \max_{\mathbf{u} \in \mathcal{Q}_d} \langle \mathbf{u}, \mathbf{W}\mathbf{x} \rangle - \mu p_d(\mathbf{u}) \ , \tag{3.39}$$

dove $p_d(\mathbf{u})$ è una funzione di prossimità per $\mathcal{Q}_d$, cioè $p_d(\mathbf{u})$ è una funzione fortemente convessa, con parametro di convessità σ_d, inoltre si può imporre che p_d si annulli in un qualche punto di $\mathcal{Q}_d$.

Nesterov dimostra che f_μ è differenziabile, e che il suo gradiente risulta [3]:

$$\nabla f_\mu(\mathbf{x}) = \mathbf{W}^* \mathbf{u}_\mu(\mathbf{x}) \ , \tag{3.40}$$

dove $\mathbf{u}_\mu(\mathbf{x})$ indica la soluzione ottima del problema (3.39). Inoltre è stato dimostrato che ∇f_μ risulta essere lipschitziano con costante:

$$L_\mu = \frac{1}{\mu \sigma_d} \|\mathbf{W}\|^2 \ . \tag{3.41}$$

Con questa approssimazione è dunque possibile applicare il metodo di Nesterov come proposto in sezione 3.5. Per un dato valore di μ la velocità di convergenza dell'algoritmo è dell'ordine di $\mathcal{O}(1/k^2)$. Se si descrive la convergenza in termini di numero di iterazioni necessarie per raggiungere un data precisione ϵ sulla soluzione, cioè il numero di iterazioni per trovare una soluzione x tale che $|f_\mu(\mathbf{x}) - \min f_\mu| < \epsilon$, si nota che, dato che μ è proporzionale alla precisione dell'approssimazione, e L_μ è proporzionale a $1/\mu \approx 1/\epsilon$, la velocità di convergenza è dell'ordine di $\mathcal{O}(\sqrt{L_\mu/\epsilon}) \approx \mathcal{O}(1/\epsilon)$, mentre le normali tecniche a sub–gradiente hanno velocità di convergenza dell'ordine di $\mathcal{O}(1/\epsilon^2)$.

3.6 Algoritmo NESTA

NESTA è un algoritmo che utilizza ed estende i risultati della teoria di Nesterov per poter essere applicato nell'ambito del Compressive Sensing (CS). Più precisamente, la tecnica CS comunemente si basa sulla risoluzione di un problema di minimizzazione vincolata della norma ℓ_1, in formule:

$$\begin{aligned} &\min \|\mathbf{x}\|_{\ell_1} \ , \\ &\|\mathbf{b} - \mathbf{A}\mathbf{x}\|_{\ell_2} \leq \epsilon \ , \end{aligned} \tag{3.42}$$

[3] con il simbolo * viene indicata l'operazione di trasposizione

dove $\mathbf{x} \in \mathbb{R}^n$ è un vettore assunto sparso (tale che solo k componenti siano significativamente diverse da zero, con $k \ll n$), $\mathbf{A} \in \mathbb{R}^{m \times n}$ è una matrice singolare a righe indipendenti con $k < m < n$. $\mathbf{A}^T\mathbf{A}$ è dunque il proiettore sullo spazio vettoriale generato dalle righe della matrice $\mathbf{A}$; per ora si assuma che $\mathbf{A}^T\mathbf{A}$ sia un proiettore ortogonale, cioè che le righe di $\mathbf{A}$ siano ortonormali.

Come visto, l'algoritmo di Nesterov si applica a funzioni regolari (più precisamente a funzioni il cui gradiente sia lipschitziano). Nel caso di una funzione non regolare, come $\|\cdot\|_{\ell_1}$, il metodo è applicabile se si esprime $\|\cdot\|_{\ell_1}$ nella forma:

$$\|\mathbf{x}\|_{\ell_1} = \max_{\mathbf{u} \in Q_d} \langle \mathbf{u}, \mathbf{x} \rangle \ , \tag{3.43}$$

dove l'insieme Q_d è la palla di centro l'origine e raggio unitario definita dalla norma ℓ_∞, cioè:

$$Q_d = \{\mathbf{u} : \|\mathbf{u}\|_\infty \leq 1\} \ . \tag{3.44}$$

Non essendo derivabile nell'origine, la funzione $\|\cdot\|_{\ell_1}$ non è regolare, e il suo gradiente non è continuo. Usando la forma (3.43) è possibile approssimare $\|\cdot\|_{\ell_1}$ con la funzione regolare:

$$f_\mu(\mathbf{x}) = \max_{\mathbf{u} \in Q_d} \{\langle \mathbf{u}, \mathbf{x} \rangle - \mu p_d(\mathbf{u})\} \ , \tag{3.45}$$

dove $p_d(\mathbf{u})$ è la funzione di prossimità (*prox–function*). Nesterov ha dimostrato [25] che se $p_d(\mathbf{u})$ è una funzione fortemente convessa e tale per cui si annulli in un qualche punto $\mathbf{u}_d^c \in Q_d$ allora f_μ risulta regolare, il suo gradiente lipschitziano e dunque il metodo si può applicare. Scegliendo:

$$p_d(\mathbf{u}) = \frac{1}{2}\|\mathbf{u}\|_{\ell_2}^2 \ , \tag{3.46}$$

di fatto si approssima $\|\cdot\|_{\ell_1}$ con f_μ che risulta essere la funzione di Huber. Infatti calcolando la derivata secondo ogni componente u_i si ha:

$$\frac{\partial f_\mu}{\partial u_i} = x_i - \mu u_i \ . \tag{3.47}$$

Il segno della derivata è positivo se $u_i \leq \frac{x_i}{\mu}$ e negativo per $u_i > \frac{x_i}{\mu}$. Nel punto $u_i = \frac{x_i}{\mu}$ si ha dunque il massimo della funzione f_μ. Ricordando ora che $\mathbf{u} \in Q_d$, quindi $|u_i| \leq 1 \quad \forall i$ si possono distinguere tre casi:

- $|x_i| \leq \mu$
  in questo caso risulta $\left|\frac{x_i}{\mu}\right| \leq 1$ il punto di massimo è proprio in $u_i = \frac{x_i}{\mu}$ e il massimo vale $\frac{x_i^2}{2\mu}$, ossia [4] $f_\mu(\mathbf{x})[i] = \frac{x_i^2}{2\mu}$;
- $|x_i| > \mu$ e $x_i > 0$
  in questo caso nell'intervallo $-1 \leq u_i \leq 1$ la derivata risulta sempre positiva, il punto di massimo è perciò $u_i = 1$ e il massimo vale $x_i - \frac{\mu}{2}$, ossia $f_\mu(\mathbf{x})[i] = x_i - \frac{\mu}{2}$;

[4] Con la notazione $f_\mu(\mathbf{x})[i]$ viene indicata la funzione f_μ calcolata per la componente i–esima di $\mathbf{x}$.

- $|x_i| > \mu$ e $x_i < 0$
  in questo caso invece la derivata risulta sempre negativa in $-1 \leq u_i \leq 1$, il punto di massimo è perciò $u_i = -1$ e il massimo vale $-x_i - \frac{\mu}{2}$, ossia $f_\mu(\mathbf{x})[i] = -x_i - \frac{\mu}{2}$;

Il ragionamento si può ripetere per ogni componente. Riassumendo, per ogni componente di f_μ si ha:

$$f_\mu(\mathbf{x})[i] = \begin{cases} \dfrac{x_i^2}{2\mu} & |x_i| < \mu \\ |x_i| - \dfrac{\mu}{2} & \text{altrimenti} \end{cases} , \tag{3.48}$$

e il gradiente di f_μ ha la componente i uguale a:

$$\nabla f_\mu(\mathbf{x})[i] = \begin{cases} \dfrac{x_i}{\mu} & |x_i| < \mu \\ \operatorname{sgn}(x_i) & \text{altrimenti} \end{cases} . \tag{3.49}$$

Da cui si vede, come previsto, che ∇f_μ è lipschitziano con costante $L_\mu = \frac{1}{\mu}$, da cui la regolarità di f_μ, condizione necessaria per applicare il metodo di Nesterov.

Il problema (3.42) si approssima dunque con il seguente:

$$\begin{aligned} \min f_\mu(\mathbf{x}) \ , \\ \|\mathbf{b} - \mathbf{A}\mathbf{x}\|_{\ell_2} \leq \epsilon \ . \end{aligned} \tag{3.50}$$

All'iterazione k-esima, una volta calcolata f_μ e ∇f_μ per un punto $\mathbf{x}_k \in \mathbb{R}^n$, i passi successivi dell'algoritmo di Nesterov richiedono di aggiornare le due successioni $\{\mathbf{y}_k\}$ e $\{\mathbf{z}_k\}$.

3.6.1 Aggiornamento di $\mathbf{y}_k$

$\mathbf{y}_k$ è definito come soluzione del problema vincolato:

$$\begin{aligned} \mathbf{y}_k = \arg\min_{\mathbf{x}} \frac{L_\mu}{2}\|\mathbf{x}_k - \mathbf{x}\|_{\ell_2}^2 + \langle \nabla f_\mu(\mathbf{x}_k), \mathbf{x} - \mathbf{x}_k \rangle \ , \\ \|\mathbf{b} - \mathbf{A}\mathbf{x}\|_{\ell_2} \leq \epsilon \ , \end{aligned} \tag{3.51}$$

che è un problema di ricerca del punto di minimo di una funzione convessa.

$\mathbf{y}_k$ si può determinare usando il metodo dei moltiplicatori di Lagrange; a tal fine si scriva la Lagrangiana:

$$\mathcal{L}(\mathbf{x}, \lambda) = \frac{L_\mu}{2}\|\mathbf{x}_k - \mathbf{x}\|_{\ell_2}^2 + \langle \nabla f_\mu(\mathbf{x}_k), \mathbf{x} - \mathbf{x}_k \rangle + \frac{\lambda}{2}(\|\mathbf{b} - \mathbf{A}\mathbf{x}\|_{\ell_2}^2 - \epsilon^2) \ , \tag{3.52}$$

e le corrispondenti condizioni KKT:

$$L_\mu(\mathbf{y}_k - \mathbf{x}_k) + \nabla f_\mu(\mathbf{x}_k) + \lambda \mathbf{A}^*(\mathbf{A}\mathbf{y}_k - \mathbf{b}) = 0 \ , \tag{3.53}$$

$$\|\mathbf{b} - \mathbf{A}\mathbf{y}_k\|_{\ell_2} \leq \epsilon \ , \tag{3.54}$$

$$\lambda \geq 0 \ , \tag{3.55}$$

$$\lambda(\|\mathbf{b} - \mathbf{A}\mathbf{y}_k\|_{\ell_2}^2 - \epsilon^2) = 0 \ . \tag{3.56}$$

Dalla condizione di stazionarietà (3.53) si può ricavare direttamente $\mathbf{y}_k$, nella forma (vedi Appendice A):

$$\mathbf{y}_k = \left(I - \frac{\lambda}{\lambda + L_\mu} \mathbf{A}^* \mathbf{A} \right) \left(\frac{\lambda}{L_\mu} \mathbf{A}^* \mathbf{b} + \mathbf{x}_k - \frac{1}{L_\mu} \nabla f_\mu(\mathbf{x}_k) \right) . \tag{3.57}$$

Il valore del moltiplicatore lagrangiano λ si ricava direttamente dalla condizione di complementarietà (3.56) (vedi Appendice B):

$$\lambda = \max \left(0, \frac{L_\mu}{\epsilon} \left\| \mathbf{b} - \mathbf{A} \left(\mathbf{x}_k - \frac{1}{L_\mu} \nabla f_\mu(\mathbf{x}_k) \right) \right\|_{\ell_2} - 1 \right) . \tag{3.58}$$

3.6.2 Aggiornamento di $\mathbf{z}_k$

Analogamente $\{\mathbf{z}_k\}$ è definito come soluzione del problema vincolato:

$$\mathbf{z}_k = \arg\min_{\mathbf{x}} \frac{L_\mu}{\sigma_p} p_p(\mathbf{x}) + \left\langle \sum_{i \leq k} \alpha_i \nabla f_\mu(\mathbf{x}_i), \mathbf{x} - \mathbf{x}_k \right\rangle , \tag{3.59}$$

$$\|\mathbf{b} - \mathbf{A}\mathbf{x}\|_{\ell_2} \leq \epsilon .$$

La funzione $p_p(\mathbf{x})$ è la funzione di prossimità primale (primal prox–function); p_p è regolare regolare, convessa (con parametro σ_p) e auspicabilmente dovrebbe avere un minimo corrispondente, o molto vicino, alla soluzione ottima del problema (3.50). La scelta di Candès in [26] è:

$$p_p(\mathbf{x}) = \frac{1}{2} \|\mathbf{x} - \mathbf{x}_0\|_{\ell_2}^2 , \tag{3.60}$$

dove $\mathbf{x}_0 \in \mathbb{R}^n$ è una stima iniziale della soluzione ottima del problema (3.50).

Va notato che la differenza tra $\{\mathbf{y}_k\}$ e $\{\mathbf{z}_k\}$ è che quest'ultima viene calcolata a partire dalla somma pesata dei gradienti $\sum_{i \leq k} \alpha_i \nabla f_\mu(\mathbf{x}_i)$. Questa procedura fa si che $\{\mathbf{z}_k\}$ tenda ad avere un comportamento meno oscillatorio di $\{\mathbf{y}_k\}$, tenga in considerazione le iterazioni passate e permetta di non allontanarsi troppo da $\mathbf{x}_0$.

Un'altra osservazione da fare è che all'iterazione k la velocità di convergenza è proporzionale a $p_p(\mathbf{x}_k)$, quindi una buona scelta di $\mathbf{x}_0$ rende l'algoritmo più veloce.

Il problema (3.59) si risolve seguendo gli stessi passi fatti per $\mathbf{y}_k$, a partire dalla lagrangiana:

$$\mathcal{L}(\mathbf{x}, \lambda) = \frac{L_\mu}{2} \|\mathbf{x} - \mathbf{x}_0\|_{\ell_2}^2 + \left\langle \sum_{i \leq k} \alpha_i \nabla f_\mu(\mathbf{x}_i), \mathbf{x} - \mathbf{x}_k \right\rangle + \frac{\lambda}{2} (\|\mathbf{b} - \mathbf{A}\mathbf{x}\|_{\ell_2}^2 - \epsilon^2) , \tag{3.61}$$

e dalle condizioni KKT, che devono valere nel punto di minimo:

$$L_\mu(\mathbf{z}_k - \mathbf{x}_0) + \sum_{i \leq k} \nabla f_\mu(\mathbf{x}_i) + \lambda \mathbf{A}^*(\mathbf{A}\mathbf{z}_k - \mathbf{b}) = 0 , \tag{3.62}$$

$$\|\mathbf{b} - \mathbf{A}\mathbf{z}_k\|_{\ell_2} \leq \epsilon , \tag{3.63}$$

$$\lambda \geq 0 , \tag{3.64}$$

$$\lambda(\|\mathbf{b} - \mathbf{A}\mathbf{z}_k\|_{\ell_2}^2 - \epsilon^2) = 0 . \tag{3.65}$$

Procedendo come mostrato in Appendice B, si ottiene:

$$\mathbf{z}_k = \left(I - \frac{\lambda}{\lambda + L_\mu} \mathbf{A}^* \mathbf{A} \right) \left(\frac{\lambda}{L_\mu} \mathbf{A}^* \mathbf{b} + \mathbf{x}_0 - \frac{1}{L_\mu} \sum_{i \leq k} \alpha_i \nabla f_\mu(\mathbf{x}_i) \right) , \tag{3.66}$$

dove il valore del moltiplicatore lagrangiano è dato da:

$$\lambda = \max \left(0, \frac{L_\mu}{\epsilon} \left\| \mathbf{b} - \mathbf{A} \left(\mathbf{x}_0 - \frac{1}{L_\mu} \sum_{i \leq k} \nabla f_\mu(\mathbf{x}_i) \right) \right\|_{\ell_2} - 1 \right) . \tag{3.67}$$

3.6.3 Aggiornamento di $\mathbf{x}_k$

Una volta calcolati $\mathbf{y}_k$ e $\mathbf{z}_k$ a partire da $\mathbf{x}_k$ e $\nabla f_\mu(\mathbf{x}_k)$, se non è verificata una condizione di arresto, si procede all'iterazione successiva dell'algoritmo, calcolando il punto $\mathbf{x}_{k+1}$ come segue:

$$\mathbf{x}_{k+1} = \tau_k \mathbf{z}_k + (1 - \tau_k) \mathbf{y}_k . \tag{3.68}$$

I coefficienti τ_k, così come gli α_k in (3.59), sono scelti in modo da rispettare le condizioni indicate da Nesterov in [25]. In particolare per $k \geq 0$:

$$\alpha_k = \frac{k+1}{2} , \tag{3.69}$$

$$\tau_k = \frac{2}{k+3} . \tag{3.70}$$

3.6.4 Condizione di arresto

Ad ogni iterazione dell'algoritmo viene verificata una condizione per decidere se la soluzione corrente è da considerare ottima o se è necessario iterare ulteriormente l'algoritmo. Un criterio semplice e naturale per fermare l'algoritmo è quello di verificare la variazione relativa della soluzione $f_\mu(\mathbf{x}_k)$. Si definisca:

$$\Delta f_\mu \triangleq \frac{\left| f_\mu(\mathbf{x}_k) - \bar{f}_\mu(\mathbf{x}_k) \right|}{\bar{f}_\mu(\mathbf{x}_k)} , \tag{3.71}$$

dove:

$$\bar{f}_\mu(\mathbf{x}_k) \triangleq \frac{1}{\min\{10, k\}} \sum_{t=1}^{\min\{10,k\}} f_\mu(\mathbf{x}_{k-t}) , \tag{3.72}$$

è la media delle soluzioni trovate al termine delle dieci iterazioni più recenti.

L'algoritmo viene fermato se:

$$\Delta f_\mu < \delta ,$$

per qualche $\delta > 0$. Anche la scelta di tale parametro, assieme a quella del parametro μ spiegato in sezione 3.6.5, influisce sulla precisione della soluzione ottima.

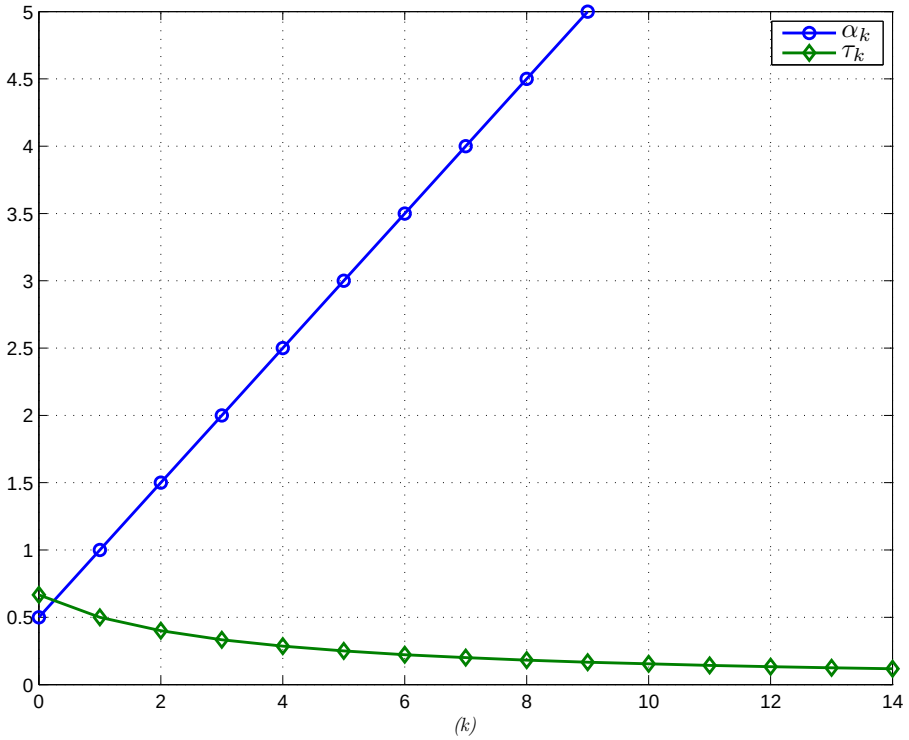


Figura 3.11: Andamento dei parametri α_k e τ_k.

3.6.5 Parametro μ

La scelta del parametro μ deve essere fatta in modo da trovare un compromesso tra la bontà dell'approssimazione regolare f_μ e la velocità di convergenza dell'algoritmo. Vale infatti:

$$\lim_{\mu \to 0} f_\mu(\mathbf{x}) = \|\mathbf{x}\|_{\ell_1} \ .$$

Mentre, come già visto, la velocità di convergenza è proporzionale a μ^{-1}:

$$f(\mathbf{x}_k) - f(\mathbf{x}^\star) \leq \frac{4p_p(\mathbf{x}^\star)}{\mu(k+1)^2} \ .$$

Va notato che, come riportato in [26, 27], se non si tiene in considerazione la possibilità di avere dati rumorosi, μ è direttamente legato alla precisione dell'algoritmo. Se il segnale da recuperare è esattamente sparso, nella soluzione calcolata dall'algoritmo l'errore sulle componenti a zero è dell'ordine di μ.

3.6.6 Complessità computazionale

In problemi di grandi dimensioni ($n \gg 1$) la maggior parte della complessità è associata all'applicazione degli operatori $\mathbf{A}$ e $\mathbf{A}^*$. Se si indica con $\mathcal{C}_A$ il costo per un'operazione che coinvolga $\mathbf{A}$ o $\mathbf{A}^*$, è possibile calcolare la complessità computazionale di ogni iterazione dell'algoritmo. Il primo passo dell'algoritmo, ovvero il calcolo di ∇f_μ, richiede solo operazioni tra vettori, dell'ordine di $\mathcal{O}(n)$, mentre aggiornare $\mathbf{y}_k$ e $\mathbf{z}_k$ richiede di applicare $\mathbf{A}$ (o $\mathbf{A}^*$) 6 volte (il valore di $\mathbf{A}^*\mathbf{b}$

può essere calcolato una sola volta). Quindi la complessità totale di una singola iterazione dell'algoritmo è $6\mathcal{C}_A + \mathcal{O}(n)$.

Nelle applicazioni specifiche del metodo di Nesterov per problemi di compressive sensing la complessità può essere ridotta. Infatti, con CS solitamente la matrice $\mathbf{A}$ è ricavata dal campionamento di una trasformazione unitaria $\mathbf{U}$, che consente di sfruttare metodi rapidi per il calcolo di prodotti tra matrice e vettori. Se $\mathbf{A} = \mathbf{RU}$, dove $\mathbf{R}$ è una matrice ottenuta da una permutazione e un campionamento della matrice identità, con un solo valore a 1 per ogni riga e al più un 1 per colonna, si può operare la minimizzazione direttamente nel dominio $\mathbf{U}$, grazie al cambio di variabile $\mathbf{x} \leftarrow \mathbf{Ux}$. Il nuovo problema diventa:

$$\min \hat{f}_\mu(\mathbf{x}) \ ,$$

$$\|\mathbf{b} - \mathbf{Rx}\|_{\ell_2} \leq \epsilon \ ,$$

dove $\hat{f}_\mu = f_\mu \circ \mathbf{U}^*$. Il gradiente della funzione composta è:

$$\nabla \hat{f}_\mu(\mathbf{x}) = \mathbf{U} \nabla f_\mu(\mathbf{U}^* \mathbf{x}) \ .$$

Con il cambio di variabile l'aggiornamento di $\mathbf{y}_k$ e $\mathbf{z}_k$ non richiede l'applicazione di $\mathbf{U}$, infatti risulta:

$$\mathbf{y}_k = \left(I - \frac{\lambda}{\lambda + L_\mu} \mathbf{R}^* \mathbf{R} \right) \left(\frac{\lambda}{L_\mu} \mathbf{R}^* \mathbf{b} + \mathbf{x}_k - \frac{1}{L_\mu} \nabla f_\mu(\mathbf{x}_k) \right) \ , \tag{3.73}$$

con:

$$\lambda = \max \left(0, \frac{L_\mu}{\epsilon} \left\| \mathbf{b} - \mathbf{R} \left(\mathbf{x}_k - \frac{1}{L_\mu} \nabla \hat{f}_\mu(\mathbf{x}_k) \right) \right\|_{\ell_2} - 1 \right) \ , \tag{3.74}$$

e la complessità dell'aggiornamento delle due successioni $\mathbf{y}_k$ e $\mathbf{z}_k$ si riduce a $\mathcal{O}(n)$. Se si indica con $\mathcal{C}_U$ il costo dell'applicazione di $\mathbf{U}$ o $\mathbf{U}^*$, il primo passo dell'algoritmo ha una complessità di $2\mathcal{C}_U$, e il costo totale per iterazione si abbassa a $2\mathcal{C}_U + \mathcal{O}(n)$.

3.6.7 Continuazione

Utilizzando la tecnica informatica nota come continuazione è possibile, in caso di $n \gg 1$, migliorare notevolmente le prestazioni di NESTA in termini di velocità di convergenza. In informatica, la continuazione è un metodo che permette di salvare lo stato corrente di esecuzione di un programma per poi riprendere l'esecuzione a partire da questo stato in un momento successivo.

Questa tecnica si rivela utile per risolvere un problemi dove la precisione richiesta è elevata (e di conseguenza il tempo richiesto dall'algoritmo aumenta); infatti per ottenere una soluzione più rapidamente è possibile andare a risolvere una serie di problemi, partendo da una precisione minore e servendosi dei risultati di ogni problema (lo stato corrente) come punto di partenza per quello successivo, in cui si utilizzerà una precisione maggiore. Proseguendo in questo modo per T volte, fino a raggiungere la precisione desiderata, si migliora la velocità dell'algoritmo.

Si ricorda che per l'aggiornamento di $\mathbf{y}_k$ (e lo stesso vale per $\mathbf{z}_k$) si deve risolvere un problema del tipo:

$$\begin{aligned}\mathbf{y}_k &= \arg\min_{\mathbf{x}\in\mathcal{Q}_p} \frac{L_\mu}{2}\|\mathbf{x}-\mathbf{x}_k\|_{\ell_2}^2 + \langle \mathbf{c}, \mathbf{x}\rangle \\ &= \arg\min_{\mathbf{x}\in\mathcal{Q}_p} \left\|\mathbf{x} - \left(\mathbf{x}_k - \frac{\mathbf{c}}{L_\mu}\right)\right\|_{\ell_2}^2 ,\end{aligned}$$

dove $\mathcal{Q}_p = \{\mathbf{x} | \|\mathbf{A}\mathbf{x}-\mathbf{b}\|_{\ell_2} \leq \epsilon\}$. Il problema è simile a quello del gradiente proiettato dove L_μ^{-1} ha il ruolo di passo [6]. Dato che L_μ^{-1} è direttamente proporzionale a μ, maggiore è μ, maggiore è il passo e più rapida è la convergenza.

Altre due osservazioni vanno fatte ricordando l'espressione ricavata per il tasso di convergenza dell'algoritmo, ossia:

$$f_\mu(\mathbf{y}_k) - f_\mu(\mathbf{x}_\mu^\star) \leq \frac{2L_\mu\|\mathbf{x}_\mu^\star - \mathbf{x}_0\|_{\ell_2}^2}{k^2} ,$$

dove $\mathbf{x}_\mu^\star$ è la soluzione ottima di f_μ. Come si vede, il tasso di convergenza è proporzionale μ^{-1}, quindi un valore alto di μ permette all'algoritmo di convergere più velocemente. D'altra parte scegliere $\mathbf{x}_0$ in modo che sia vicino alla soluzione ottima $\mathbf{x}_\mu^\star$, fa si che $p_p(\mathbf{x}_\mu^\star) = \frac{1}{2}\|\mathbf{x}_\mu^\star - \mathbf{x}_0\|_{\ell_2}^2$ sia un valore piccolo, con conseguente miglioramento della precisione e del tasso di convergenza.

Queste osservazioni motivano il seguente algoritmo che utilizza la continuazione:

Inizializzare μ_0, $\mathbf{x}_0$ e il numero di iterazioni della continuazione T. Per $t \geq 1$,

1. Applicare l'algoritmo di Nesterov con $\mu = \mu^{(t)}$ e $\mathbf{x}_0 = \mathbf{x}_{\mu^{t-1}}$;
2. Ridurre il valore di μ : $\mu^{(t+1)} = \gamma\mu^{(t)}$ con $\gamma < 1$;

Arrestare l'algoritmo quando viene raggiunto un certo valore di precisione μ_f, scelto all'inizio, ossia quando $\mu^{(t+1)} > \mu_f$.

4 Risultati simulativi per l'algoritmo di Nesterov

Per testare validità e prestazioni del metodo di Nesterov sono state effettuate alcune simulazioni del processo di recupero dei dati attraverso la tecnica del CS. Nelle prossime sezioni verrà descritto lo scenario e il tipo di segnali usati per le simulazioni, poi si riporteranno alcuni risultati al variare dei parametri dell'algoritmo e infine si metteranno a confronto le prestazioni dell'algoritmo NESTA con quelle di altri algoritmi di minimizzazione presenti in letteratura.

4.1 Scenario simulazioni

Le simulazioni sono state eseguite utilizzando dei segnali reali, ottenuti da tre campagne di letture della rete di sensori wireless (WSN) presso il Department of Information Engeneering, Università di Padova. La rete consiste di $N = 68$ sensori TmoteSky, ognuno dei quali effettua la misura di cinque segnali diversi (Temperatura, Umidità, Voltaggio e Luminosità su due diversi range di lunghezze d'onda, $320 - 730nm$ e $320 - 1100nm$). Ogni campagna consiste nella collezione delle letture di un certo numero di sensori effettuate ogni 5 minuti per 3 giorni consecutivi.

Il segnale di interesse viene rappresentato da un vettore che contiene le misurazioni degli N sensori ad ogni istante temporale, $\mathbf{x}^{(k)} \in \mathbb{R}^N$, dove k è il tempo discreto in cui varia il segnale. In generale i segnali reali non sono stazionari, per cui la loro descrizione statistica varia nel tempo.

Il framework utilizzato per le simulazioni consiste nell'applicazione della tecnica CS utilizzando la Principal Component Analysis per studiare la struttura dei segnali e costruire le matrici di trasformazione necessarie, alternando due fasi di funzionamento:

1. una *fase di training* di $K = 2$ round durante i quali si raccolgono i dati di ogni sensore per calcolare le statistiche di interesse, cioè la media del vettore $\mathbf{x}^{(k)}$, $\bar{\mathbf{x}}$ e la sua matrice di covarianza $\hat{\mathbf{\Sigma}}$, come indicato nell'equazione 2.14.

2. una *fase di monitoraggio* di $\zeta K = 8$ round durante i quali i sensori trasmettono con una probabilità $p_{TX} < 1$ fissata, per cui in media solo $L \simeq p_{TX}N$ sensori trasmettono il proprio dato. Il segnale completo viene ricostruito

usando le statistiche calcolate in precedenza: è in questo punto che entra in gioco l'algoritmo di minimizzazione, che nel nostro caso è l'algoritmo di Nesterov.

Le prestazioni dell'algoritmo sono state valutate sulla base dell'errore sul segnale ricostruito, facile da calcolare in quanto si dispone del segnale originale definito in Eq (2.4) e qui riportato per comodità:

$$\xi_R^{(k)} = \frac{\|\mathbf{x^{(k)}} - \hat{\mathbf{x}}^{(\mathbf{k})}\|_2}{\|\mathbf{x}^{(k)}\|_2} \,, \tag{4.1}$$

dove $\hat{\mathbf{x}}^{(k)}$ è la ricostruzione del segnale originario $\mathbf{x}^{(k)}$ ottenuta dal segnale compresso $\mathbf{y}^{(k)}$. Nell'asse delle ascisse dei grafici che seguono si è preferito riportare la probabilità di trasmissione come parametro di riferimento, ovvero il rapporto tra il numero di sensori che trasmettono il proprio dato e il numero totale di sensori che compongono la rete, piuttosto che il costo in termini di numero di trasmissioni, dato che in ogni campagna di lettura il numero di sensori utilizzati è diverso.

4.2 Prestazioni NESTA

Le prestazioni dell'algoritmo di Nesterov sono state valutate variando alcuni parametri di funzionamento del metodo stesso. In sostanza NESTA risolve una versione regolarizzata di un problema di minimizzazione in norma ℓ_1 del tipo:

$$\min f_\mu(\mathbf{x}) \,, \tag{4.2}$$

$$\|\mathbf{Ax} - \mathbf{b}\|_{\ell_2} \leq \epsilon \,, \tag{4.3}$$

dove f_μ è l'approssimazione regolare della norma ℓ_1. Le grandezze su cui si può agire sono tre:

- μ, la precisione dell'approssimazione regolare, che è legata anche alla velocità di convergenza dell'algoritmo;
- ϵ, la tolleranza sui vincoli, che permette di rilassare a piacere i vincoli del problema;
- δ, la variabile che controlla il criterio di arresto dell'algoritmo. Va ricordato infatti che l'algoritmo si ferma quando da un'iterazione alla successiva la soluzione ha subito una variazione minore di δ.

I segnali di ingresso dell'algoritmo, sono stati divisi in due gruppi: da una parte le letture di temperatura e umidità, che sono segnali poco variabili sia nel tempo che nello spazio, dall'altra le letture della luminosità, meno costanti e quindi soggette a un errore di ricostruzione maggiore. Il segnale che riguarda il voltaggio non è stato utilizzato, in quanto di scarso interesse perchè quasi costante.

É stata applicata la tecnica CS ai segnali descritti facendo variare la probabilità di trasmissione a passi di 0.05 da una $p_{min} = 0.2$ a una $p_{max} = 1$, ed i

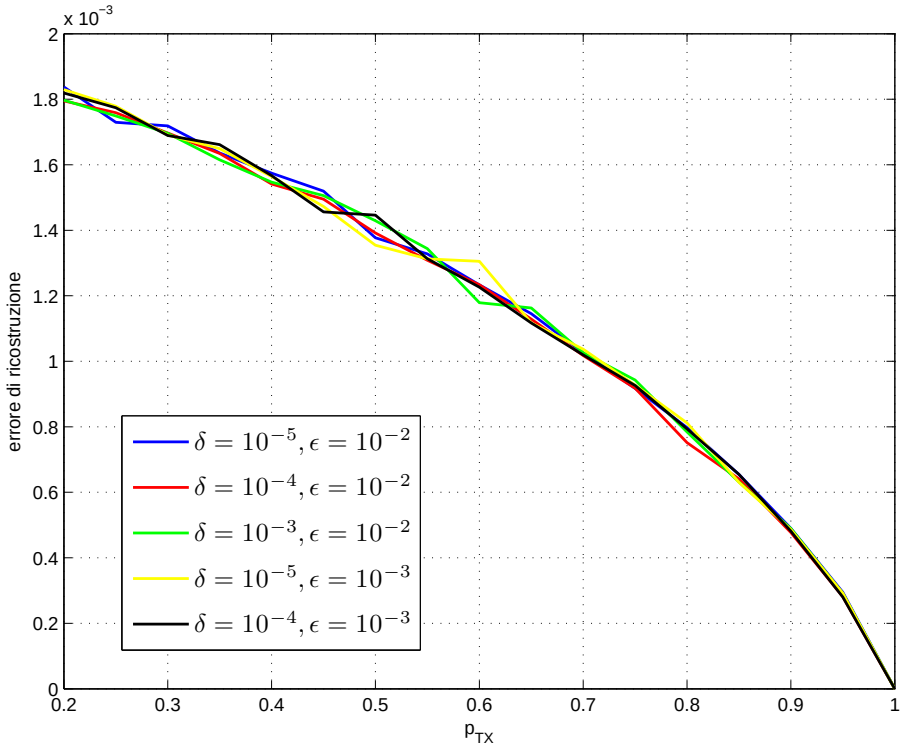


Figura 4.1: Prove effettuate variando i parametri δ e ϵ con segnali di temperatura e umidità.

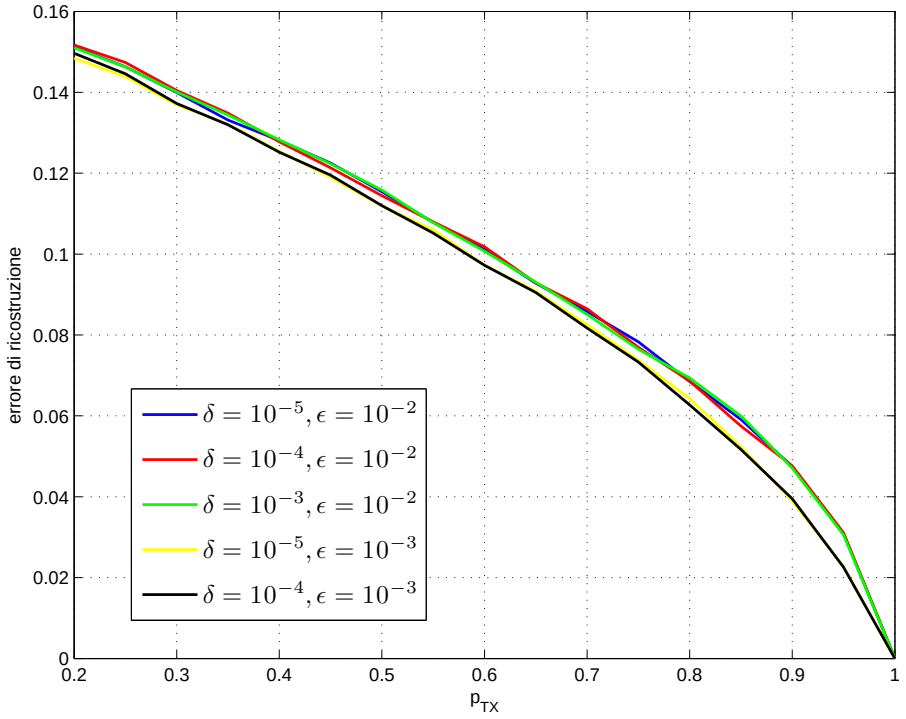


Figura 4.2: Prove effettuate variando i parametri δ e ϵ con segnali di luminosità.

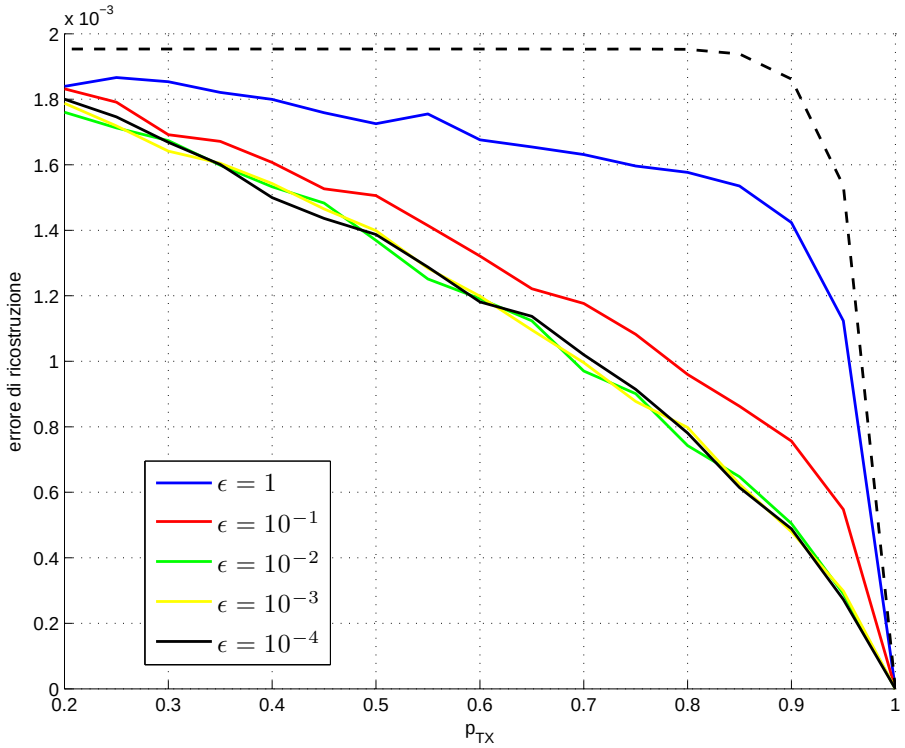


Figura 4.3: Errore di ricostruzione al variare della probabilità di trasmissione con segnali di temperatura e umidità, $\delta = 0.01$.

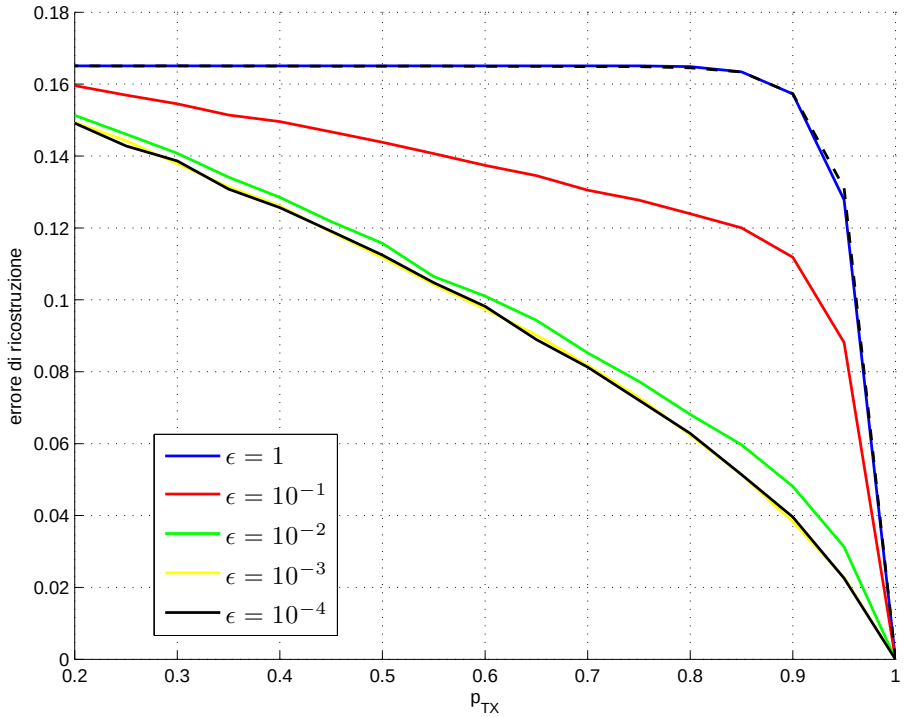


Figura 4.4: Errore di ricostruzione al variare della probabilità di trasmissione con segnali di luminosità, $\delta = 0.01$.

tre parametri sopra descritti. Per ogni scelta di p_{TX} e dei parametri sono state effettuate diverse prove e nei grafici si riporta il risultato mediato.

Si è osservato, in altre simulazioni non riportate qui, che il parametro μ non influisce significativamente sull'errore di ricostruzione dell'algoritmo, ma solo sulla velocità di convergenza, per questo motivo il valore di μ è stato mantenuto costante ($\mu = 0.01$, scelto dai risultati delle simulazioni effettuate come valore che garantisce una buona approssimazione regolare). Anche i valori che regolano la durata delle fasi di training e monitoraggio sono stati fissati. Le diverse curve dei grafici presentati sono state ottenute variando i parametri ϵ e δ, cioè la tolleranza sui vincoli rispetto al criterio di arresto dell'algoritmo.

Nelle Figure 4.1 e 4.2 sono state utilizzate diverse combinazioni di valori per i parametri ϵ e δ, i risultati, come si può vedere non differiscono significativamente.

Attraverso altre simulazioni si è verificato che un valore di δ che porta a buone prestazioni dell'algoritmo è $\delta = 0.01$, si è quindi cercato di ottimizzare il valore di ϵ, mantenendo invece costante quello di δ. Nelle Figure 4.3 e 4.4 viene mostrato come la variazione di ϵ influisca sulle prestazioni dell'algoritmo, più precisamente, quando ϵ è minore di δ o dello stesso ordine di grandezza si ottiene un errore minore (curve gialle, verdi e nere nei grafici), mentre quando ϵ è maggiore di δ l'errore aumenta (curve rosse e blu). Questo comportamento è dovuto al fatto che se la tolleranza sui vincoli è maggiore rispetto al criterio di arresto, lo spazio delle soluzioni possibili diventa più grande e la soluzione trovata dall'algoritmo risulta più "lontana" dalla soluzione ottima, in quanto l'algoritmo si ferma prima di aver ottenuto una opportuna risoluzione.

Infine la curva tratteggiata nelle figure è stata ottenuta dalla ricostruzione del segnale basata unicamente sulla media, senza cioè applicare l'algoritmo di minimizzazione per cercare la soluzione più sparsa. É interessante notare come questa curva faccia da bound alle prestazioni dell'algoritmo. L'applicazione dell'algoritmo permette infatti di migliorare la precisione della ricostruzione di base ottenuta utilizzando solo la media. Si nota inoltre che nel caso di segnali poco variabili (Figura 4.3) è possibile recuperare il segnale originale con un errore molto basso (circa 0.2 %) usando unicamente la media calcolata nelle fasi di training.

4.3 Confronto prestazioni

Per studiare le prestazioni dell'algoritmo di Nesterov sono state confrontate le prestazioni di NESTA con quelle di altri due algoritmi presenti in letteratura; il primo è un metodo Interior–Point chiamato $\ell_1 Magic$ [28], che al contrario dei principali algoritmi di programmazione lineare cerca la soluzione ottima spostandosi all'interno della regione ammissibile invece che sui bordi,mentre il secondo è un algoritmo di minimizzazione della norma ℓ_0 di tipo steepest–decent chiamato SL_0 [29], che ad ogni iterazione si sposta verso la direzione di massima discesa della funzione da minimizzare, come il metodo del gradiente.

Come in precedenza le simulazioni sono state organizzate dividendo i segnali in due classi in base alla variabilità: temperatura e umidità da una parte e i due segnali riguardanti la luminosità dall'altra. Come è possibile vedere dai grafici in Figura 4.5 e 4.6, la differenza in termini di prestazioni non è molto significativa,

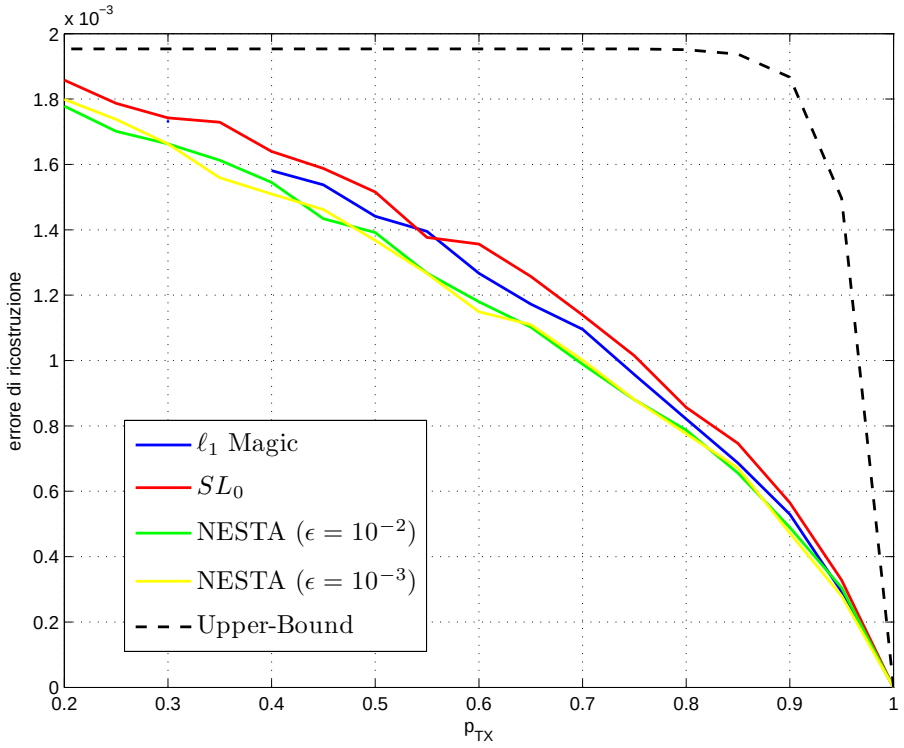


Figura 4.5: Errore di ricostruzione dei vari metodi usando segnali di temperatura e umidità.

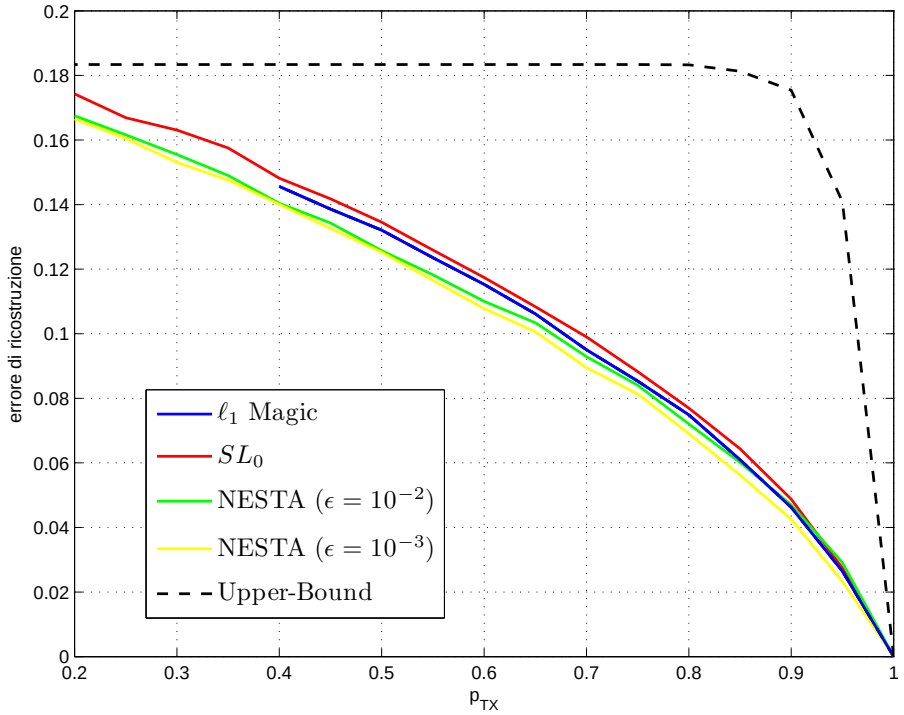


Figura 4.6: Errore di ricostruzione dei vari metodi usando segnali di luminosità.

anche se NESTA da comunque dei risultati migliori. Inoltre con l'algoritmo $\ell_1 Magic$ sono stati riscontrati dei problemi di risultati non numerici quando applicato con probabilità di trasmissione basse ($0.2 \div 0.4$).

Con riferimento alla Figura 4.5 si nota che la curva riguardante l'algoritmo $\ell_1 Magic$ si ferma per $p_{TX} < 0.4$, in quanto la funzione di calcolo usata porta a dei risultati non numerici, probabilmente per un errore nel codice. Le prestazioni dell'algoritmo NESTA risultano migliori, anche se non di molto rispetto a entrambi gli algoritmi di riferimento.

Anche in Figura 4.6 si vede come le prestazioni dell'algoritmo NESTA siano migliori rispetto a quanto si può ottenere con $\ell_1 Magic$ e SL_0, anche se le differenze risultano sempre molto contenute.

Come illustrato nei capitoli precedenti, NESTA offre la possibilità di poter essere applicato usando la tecnica della continuazione. Questa tecnica consente notevoli miglioramenti in termini di velocità di convergenza dell'algoritmo, specialmente se si vogliono spingere al limite le prestazioni (cioè usare tolleranze molto basse) e se i segnali coinvolti hanno un range dinamico molto ampio, cioè se i diversi valori in una lettura differiscono di qualche ordine di grandezza. (per segnali con 100dB di range il numero di iterazioni richieste dall'algoritmo si riduce di circa un ordine di grandezza [26]).

Nei casi esaminati però la continuazione non influisce significativamente sulle prestazioni, in quanto i segnali da ricostruire subiscono variazioni molto contenute.

5 Modulo Rebuilder

L'implementazione della tecnica Compressive Sensing e la sua integrazione nell'applicazione WSNControl sono state realizzate attraverso la scrittura di un modulo in linguaggio *C++*, chiamato *Rebuilder*, e la modifica di alcuni punti della web application stessa.

Il modulo Rebuilder, nell'architettura generale dell'applicazione, va a sostituire quelle che erano le funzionalità dei moduli di ottimizzazione e di feedback (Figura 1.3). La nuova architettura modulare dell'applicazione è rappresentata in Figura 5.1.

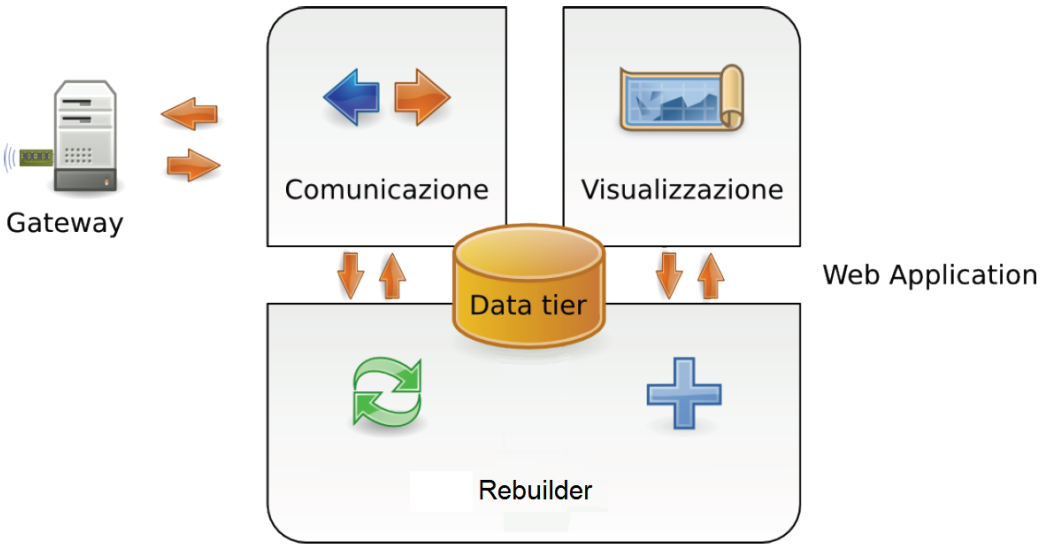


Figura 5.1: Architettura modulare dell'applicazione dopo le modifiche apportate

Entrando in un livello di dettaglio maggiore, la nuova architettura che si vuole realizzare è quella mostrata nella Figura 5.2. Considerando una rete formata da N sensori, ad ogni istante k:

- il Gateway riceve il segnale incompleto $\mathbf{y}^{(k)}$, che contiene solo $M < N$ misure, sulla base della $p_{TX}^{(k)}$ precedentemente inviata ai sensori;

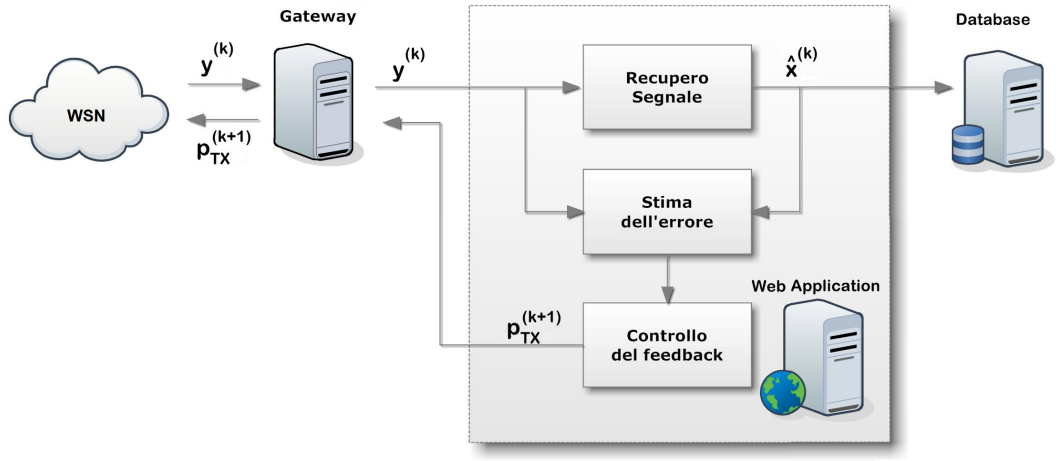


Figura 5.2: Architettura del sistema di raccolta e recupero dei dati tramite Compressive Sensing.

- il pacchetto contenente $\mathbf{y}^{(k)}$ viene inoltrato al server su cui è in esecuzione la Web Application, che attraverso il modulo Rebuilder ricostruisce il segnale completo $\hat{\mathbf{x}}^{(k)}$, formato da N misure;
- i segnali $\mathbf{y}^{(k)}$ e $\hat{\mathbf{x}}^{(k)}$ vengono utilizzati per stimare l'errore commesso nella ricostruzione, attraverso una tecnica che sarà discussa in dettaglio nella Sezione 5.3;
- in base alla stima dell'errore, il blocco di controllo del feedback calcola la $p_{TX}^{(k+1)}$ per la lettura successiva;
- questa probabilità viene comunicata alla rete di sensori seguendo il percorso inverso dei dati.

I dati ricostruiti vengono inoltre salvati nel database, e come sarà chiarito nelle prossime sezioni, verranno usati per il calcolo delle statistiche necessarie al recupero dei dati.

5.1 Modifiche apportate a WSNControl

Oltre al modulo Rebuilder, per riuscire a realizzare un'architettura come quella mostrata in Figura 5.2, sono state necessarie alcune modifiche all'applicazione Web, al software FakeGateway e al protocollo di comunicazione WAGP. Le modifiche riguardano esclusivamente la gestione dell'informazione sulla probabilità di trasmissione, che in una prima realizzazione del codice non era stata prevista. In particolare la p_{TX} viene letta dalla Web Application nei pacchetti di risposta del modulo ricostruttore, salvata in una variabile e comunicata al Gateway nell'eventuale successiva richiesta. Sono stati aggiunti quindi dei campi per contenere le probabilità di trasmissione per ogni tipo di misura nel pacchetto che l'applicazione invia al Gateway, che prima conteneva unicamente un comando di richiesta dati (GET_DATA_COMMAND).

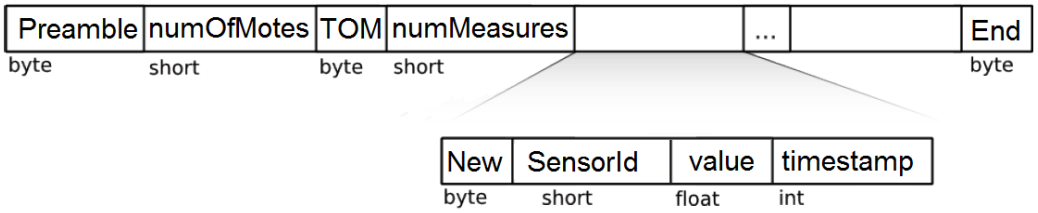


Figura 5.3: Formato del pacchetto inviato al modulo Rebuilder

Il Gateway, dal suo punto di vista, riceve le richieste e nello stesso pacchetto trova anche le probabilità di trasmissione da comunicare ai sensori. La sua funzione di forwarder quindi non viene alterata. Nel nostro caso, dove il Gateway viene simulato da un software, le p_{TX} contenute nei pacchetti di richiesta vengono usate per decidere quanti e quali valori, letti dai file, inviare nella risposta alla Web Application. Va ricordato che i valori contenuti nei file sono stati raccolti in precedenza da diversi testbed reali.

5.2 Funzionamento

Lanciata l'applicazione Web e messo in ascolto il Gateway, il modulo Rebuilder viene eseguito ogni volta che dalla rete arriva una nuova misura. All'inizio della trasmissione è necessaria una fase di inizializzazione durante la quale i sensori sono impostati per trasmettere con probabilità unitaria, così da disporre di un training set per la tecnica PCA (Principal Component Analisys). I dati, arrivati all'applicazione in un pacchetto WAGP (Figura 1.6), vengono preventivamente scritti nel database e successivamente indirizzati al modulo Rebuilder. Questo accade perchè, oltre alla misura corrente, bisogna che il modulo di ricostruzione riceva anche alcune misure precedenti.

Il formato del pacchetto con cui vengono passate le informazioni al modulo Rebuilder, rappresentato in Figura 5.3 è simile a quello del protocollo WAGP, per ogni misura di ogni sensore è sono riportati i campi *SensorId*, *value* e *timestamp*, mentre il numero totale di sensori, il numero di misure per ogni sensore e il tipo di misura sono riportati nel preambolo del pacchetto. Lo stesso formato è usato per restituire i dati alla web application una volta che sono state effettuate operazioni di ricostruzione, ma nel preambolo è riportato soltanto il valore della p_{TX} per l'iterazione successiva, come illustrato in Figura 5.4.

Una volta ricevuta la risposta, la Web application effettua alcune operazioni per verificare quali misure devono ancora essere inserite nel database.

Successivamente a questa fase di inizializzazione, la probabilità di trasmissione viene abbassata, così che il numero di sensori che trasmettono non sia più N. Se all'istante k il modulo di ricostruzione riceve una misura incompleta, esegue le seguenti operazioni:

1. legge il pacchetto in arrivo e individua il vettore $\mathbf{y}^{(k)}$ e il training set $\hat{\mathcal{T}}_K = \left\{\hat{\mathbf{x}}^{(k-K)}, \ldots, \hat{\mathbf{x}}^{(k-1)}\right\}$ usando le informazioni contenute nei timestamp;

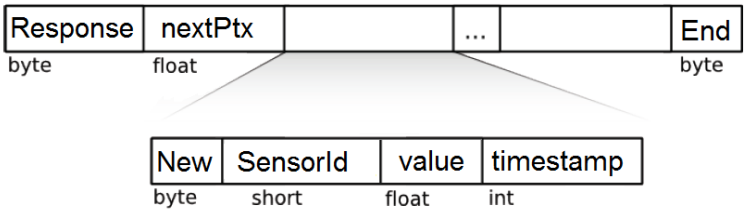


Figura 5.4: Formato del pacchetto di risposta del modulo Rebuilder

2. usa gli Id dei sensori per costruire la matrice di routing $\mathbf{\Phi}^{(k)}$;
3. costruisce la matrice di trasformazione $\mathbf{\Psi}^{(k)}$ attraverso la Principal Component Analysis applicata al training set $\hat{\mathcal{T}}_K$;
4. applica l'algoritmo di Nesterov al problema:

$$\min \|\mathbf{s}^{(k)}\|_{\ell_1} \ ,$$

$$\|\mathbf{y}^{(k)} - \mathbf{\Phi}^{(k)}\mathbf{\Psi}^{(k)}\mathbf{s}^{(k)}\|_{\ell_2} \leq \epsilon \ ;$$

5. recupera il segnale $\hat{\mathbf{x}}$ attraverso la seguente relazione:

$$\hat{\mathbf{x}}^{(k)} = \mathbf{\Psi}\mathbf{s}^{(k)} \ ;$$

6. stima l'errore commesso nella ricostruzione del segnale (Sezione 5.3);
7. sceglie la probabilità di trasmissione per l'iterazione successiva $p_{TX}^{(k+1)}$, sulla base dell'errore stimato (Sezione 5.4);
8. restituisce il segnale ricostruito all'applicazione web.

5.3 Stima dell'errore di ricostruzione

La quantità che si vuole stimare è l'errore di ricostruzione, definito in Eq. (2.4) come:

$$\xi_R^{(k)} = \frac{\|\mathbf{x}^{(k)} - \hat{\mathbf{x}}^{(k)}\|_{\ell_2}}{\|\mathbf{x}^{(k)}\|_{\ell_2}} \ , \tag{5.1}$$

dove $\hat{\mathbf{x}}^{(k)}$ è il segnale ricostruito e $\mathbf{x}^{(k)}$ il segnale originale dato da tutte le misure dei sensori. Ovviamente non si ha a disposizione $\mathbf{x}^{(k)}$, ma solo la sua versione campionata $\mathbf{y}^{(k)} = \mathbf{\Phi}^{(k)}\mathbf{x}^{(k)}$, un possibile modo di stimare l'errore è calcolare:

$$\xi_0^{(k)} = \frac{\|\mathbf{y}^{(k)} - \mathbf{\Phi}^{(k)}\hat{\mathbf{x}}^{(k)}\|_{\ell_2}}{\|\mathbf{y}^{(k)}\|_{\ell_2}} \ , \tag{5.2}$$

ma questa quantità risulta sempre nulla, in quanto i campioni ricevuti sono sempre ricostruiti perfettamente. Il metodo usato per stimare l'errore è un metodo

euristico, che si basa sulle informazioni delle ultime due iterazioni, calcolando:

$$\xi^{(k)} = \frac{\left\| \begin{bmatrix} \mathbf{y}^{(k)} \\ \mathbf{y}^{(k-1)} \end{bmatrix} - \begin{bmatrix} \mathbf{\Phi}^{(k)}\hat{\mathbf{x}}^{(k-1)} \\ \mathbf{\Phi}^{(k-1)}\hat{\mathbf{x}}^{(k)} \end{bmatrix} \right\|_{\ell_2}}{\left\| \begin{bmatrix} \mathbf{y}^{(k)} \\ \mathbf{y}^{(k-1)} \end{bmatrix} \right\|_{\ell_2}} , \tag{5.3}$$

che porta a una buona stima. É interessante notare che $\xi^{(k)}$ è influenzato non solo dell'effettivo errore compiuto nella ricostruzione, ma anche della variabilità del segnale. Se da una parte questo fatto introduce un'approssimazione nella stima dell'errore, dall'altra rende possibile il riconoscimento di variazioni improvvise del segnale, così da poter adattare la probabilità di trasmissione anche in caso di segnali irregolari.

5.4 Aggiornamento della p_{TX}

Il blocco di controllo del feedback mostrato in Figura 5.2 è responsabile del calcolo della nuova probabilità di trasmissione. Tale operazione viene effettuato mediante una tecnica AIMD (Additive Increase Multiplicative Decrease), che fa aumentare velocemente la p_{TX} se la stima dell'errore è troppo alta (sopra una soglia τ), e la fa diminuire lentamente nel caso l'errore sia contenuto.

A seconda dell'errore stimato si verifica uno di questi due casi:

- $\xi^{(k)} \geq \tau$: in questo caso l'errore è sopra la soglia, quindi è necessario aumentare velocemente la probabilità di trasmissione. Si definisce una costante $C_1 \in [1, +\infty]$ e la nuova p_{TX} viene calcolata in questo modo:

$$p_{TX}^{(k+1)} = \min \left\{ C_1 p_{TX}^{(k)}, 1 \right\} ; \tag{5.4}$$

- $\xi^{(k)} < \tau$: la stima dell'errore è sotto la soglia, quindi si può provare a diminuire lentamente la probabilità di trasmissione. Si definisce una costante $C_2 \in \{1, 2, \ldots, N\}$ e una costante p_{TX}^{min} che indica il valore minimo ammesso. La nuova p_{TX} viene ottenuta da:

$$p_{TX}^{(k+1)} = \max \left\{ p_{TX}^{(k)} - \frac{C_2}{N}, p_{TX}^{min} \right\} . \tag{5.5}$$

5.5 Dettagli d'implementazione

Nel codice scritto per realizzare il modulo di ricostruzione si è fatto uso della libreria *NewMat10* [30]. La libreria permette di dichiarari oggetti di tipo `Matrix`, `ColumnVector` e molti altri, inoltre attraverso l'overloading degli operatori, tutte le operazioni tra matrici vengono gestite automaticamente. Le funzioni offerte da NewMat10 vanno dalle semplici operazioni tra matrici, alla trasposizione, al calcolo del determinante e della matrice inversa, fino alle operazioni più complesse

come la decomposizione ai valori singolari, il calcolo di autovalori e autovettori e la Fast Fourier Transform.

La libreria è stata strutturata per le matrici di almeno 10 righe per 10 colonne, fino alla dimensione massima data dalla memoria disponibile sul calcolatore. La libreria funziona ovviamente anche per matrici piccole, ma alcune operazioni diventano piuttosto inefficienti. Inoltre alcune delle funzioni di fattorizzazione non sono ottimizzate per la memoria di paging e diventano inefficienti quando vengono utilizzate con matrici molto grandi. Per gli scopi di questa tesi, in cui si è lavorato con reti di circa un centinaio di sensori, che portano alla costruzione di matrici con 10^4 elementi, la libreria si è dimostrata efficiente e con buone prestazioni. Nel caso si volessero applicare le tecniche descritte a reti molto grandi potrebbe essere necessario utilizzare una libreria diversa, o dei metodi di calcolo ottimizzati.

6 Risultati simulativi per il framework

Il framework presentato nel capitolo precedente, realizzato per l'applicazione WSNControl, è stato utilizzato per effettuare alcune simulazioni di raccolta e ricostruzione dei dati. L'architettura utilizzata è quella descritta nel Capitolo 5 e mostrata in Figura 5.2, dove il Gateway è stato sostituito dal software FakeGateway.

I segnali utilizzati per queste simulazioni provengono da alcune campagne di lettura effettuate al Department of Information Engineering dell'Università di Padova e dalla rete LUCE (Lausanne Urban Canopy Experiment), situata nel campus dell'EPFL (Ecole Polytechnique Fédérale de Lausanne, Switzerland), e i cui dati sono disponibili online. Le due reti sono mostrate in Figura 6.1

La Tabella 6.1 riassume le caratteristiche di ogni campagna considerata. Le campagne denominate **A**, **B** e **C** si riferiscono alla rete del DEI, sono campagne in cui la dimensione della rete e relativamente ridotta (il numero di sensori è circa $N = 30$) e il numero di letture è elevato (le letture sono distribuite ad intervalli regolari di 5 minuti su un periodo di 3 giorni). Ai fini di ottenere una campagna di dimensione maggiore, in termini di numero di sensori, si è assunto che i dati delle campagne **A**, **B** e **C** fossero riferiti a 3 diverse locazioni geografiche e che i dati fossero stati raccolti negli stessi istanti di tempo. Si sono quindi unite le letture di queste tre campagne simulando così la raccolta dei dati per una rete di dimensioni pari alla somma delle dimensioni delle tre reti **A**, **B** e **C**. La campagna così ottenuta, denominata **ABC**, è stata utilizzata principalmente per testare il funzionamento dell'intero framework associato a reti di dimensioni maggiori (circa un centinaio di sensori). La campagna denominata **LUCE**, con un numero consistente di sensori, è stata utilizzata per avere una conferma delle prestazioni della tecnica Compressive Sensing applicata ad altre tipologie di segnali.

6.1 Settaggio Parametri

Per le diverse simulazioni effettuate, i parametri dell'algoritmo di Nesterov, di cui si è discusso nel Capitolo 3, sono stati scelti in accordo con i risultati ottenuti nel Capitolo 4 e mantenuti costanti. In particolare si è scelto:

(a) Rete di sensori del DEI.

(b) Rete di sensori LUCE, presso EPFL.

Figura 6.1: Disposizione dei nodi nelle reti di sensori considerate.

- $\mu = 0.01$, che garantisce un'ottima approssimazione regolare f_μ, mantenendo comunque la velocità di convergenza dell'algoritmo entro limiti accettabili;
- $\varepsilon = 0$, per fare in modo che i dati ricevuti non vengano alterati (con un valore non nullo si introduce un errore anche sui campioni ricevuti);
- $\delta = 10^{-5}$, che unitamente al valore scelto per ϵ garantisce un errore di ricostruzione basso.

Per quanto riguarda i parametri con cui si effettua l'aggiornamento della probabilità di trasmissione, dopo una fase di studio e ottimizzazione in ambiente simulativo, si sono scelti i seguenti valori:

Campagna	N° sensori	N° letture	Segnali
A	29	783	temperatura umidità luminosità
B	35	850	temperatura umidità luminosità
C	31	550	temperatura umidità luminosità
ABC	95	100	temperatura umidità luminosità
LUCE	81	200	temperatura del suolo umidità temperatura dell'ambiente

Tabella 6.1: Caratteristiche delle campagne utilizzate

- $C_1 = 1.3$, la costante moltiplicativa con cui si aumenta la p_{TX} nel caso la stima dell'errore superi la soglia τ;
- $C_2 = \{3, 5\}$, a seconda che la rete contenga un numero rispettivamente contenuto o elevato di nodi. C_2 infatti da un'indicazione media sul numero di nodi sensori che smettono di trasmettere ad ogni iterazione in cui l'errore di ricostruzione sia minore di τ;
- $\tau = 0.25$, la soglia con cui si decide se aumentare o diminuire la p_{TX}. Si è verificato che il valore scelto porta l'algoritmo a reagire velocemente a situazioni in cui i segnali subiscano variazioni improvvise, pur permettendo di sfruttare al meglio l'efficienza del metodo di ricostruzione;
- $p_{TX}^{min} = 0.20$, come limite minimo per la probabilità di trasmissione.

Si sono effettuate inoltre alcune simulazioni variando la soglia τ, per descrivere come questa incida sull'andamento medio dell'errore di ricostruzione e della probabilità di trasmissione.

6.2 Risultati

Per ogni segnale delle diverse campagne sono stati salvati l'andamento della probabilità di trasmissione e i dati ricostruiti forniti dal modulo Rebuilder ad ogni istante k. Avendo a disposizione sia i dati originali $\mathbf{x}^{(k)}$ che i dati ricostruiti $\hat{\mathbf{x}}^{(k)}$ si è potuto calcolare l'errore di ricostruzione:

$$\xi_R^{(k)} = \frac{\|\mathbf{x}^{(k)} - \hat{\mathbf{x}}^{(k)}\|_{\ell_2}}{\|\mathbf{x}^{(k)}\|_{\ell_2}} \,. \tag{6.1}$$

Si sono poi calcolati i valori medi della probabilità di trasmissione e dell'errore di ricostruzione durante le intere simulazioni. I risultati ottenuti per ogni campagna sono riportati e discussi di seguito. Per chiarezza di visualizzazione i grafici riportano l'andamento delle metriche considerate durante le prime 100 letture, mentre le medie sono valutate sulla totale durata delle simulazioni.

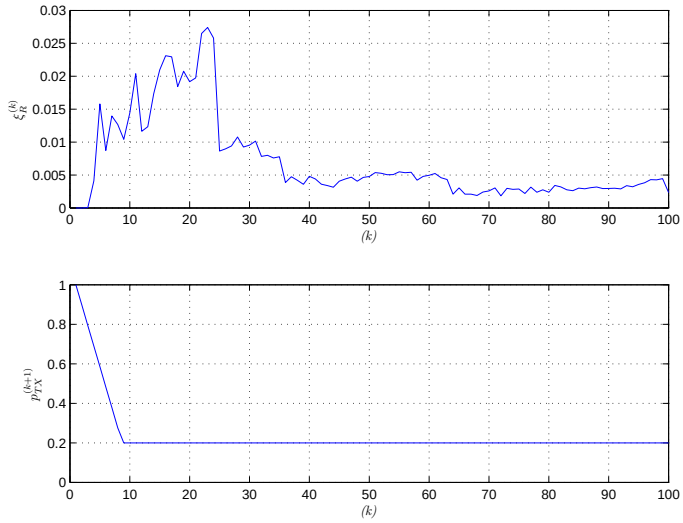
6.2.1 Campagna A

La Figura 6.2 mostra l'andamento dell'errore di ricostruzione $\xi_R^{(k)}$ e della probabilità di trasmissione scelta dal modulo per l'iterazione successiva, $p_{TX}^{(k+1)}$, per un ridotto numero di istanti temporali ($k = 1, \ldots, 100$). Gli andamenti di queste grandezze sono riportati per ognuno dei tre tipi di segnale di cui è composta la campagna. Come si può notare dai grafici, il modulo di ricostruzione risponde in modo diverso a seconda del tipo di segnale considerato.

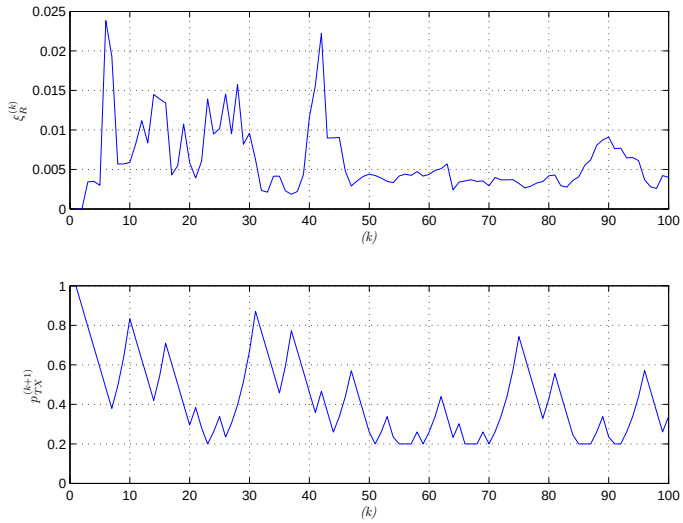
Per il segnale di temperatura Figura 6.2(a), che è un segnale poco variabile, sia nel tempo che nello spazio, il modulo Rebuilder sfrutta al massimo le possibilità del Compressive Sensing. La p_{TX} viene infatti ridotta dopo poche iterazioni al valore minimo impostato, e non subisce ulteriori variazioni. Questo perchè la stima dell'errore di ricostruzione non supera mai la soglia di 0.25. Una conferma di questo fatto è data dall'andamento dell'errore di ricostruzione, riportato nel grafico, che ad eccezione di qualche oscillazione si mantiene sempre intorno allo 0.5% (durante l'intera simulazione).

L'andamento della probabilità di trasmissione risulta molto interessante nel caso del segnale di umidità, Figura 6.2(b). Si nota infatti come questa venga continuamente adattata dal modulo di ricostruzione, che quando può ottenere una buona stima del segnale con pochi campioni la abbassa, mentre quando la variabilità del segnale e di conseguenza l'errore di ricostruzione crescono, la fa aumentare, ottenendo così un maggior numero di campioni nelle letture successive. Anche in questo caso si nota come l'errore di ricostruzione sia contenuto, con qualche picco del 2.5%, ma in media abbondantemente sotto l'1%.

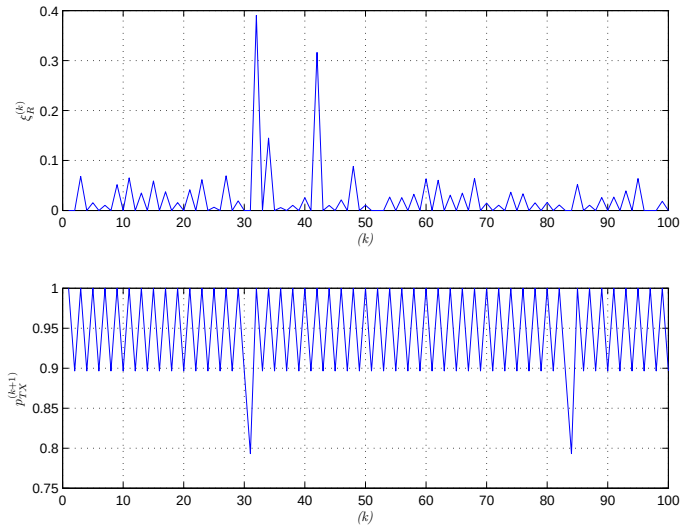
L'ultimo segnale considerato, cioè quello di luminosità, ha un comportamento diverso dai precedenti. Sia la correlazione spaziale che quella temporale sono minori, basti pensare alla differenza dei valori misurati da un sensore che si trova vicino ad una finestra piuttosto che in un corridoio o in una stanza buia, o alla differenza dei valori misurati durante il giorno e quelli raccolti durante la notte (la campagna **A** infatti raccoglie i dati monitorati in tre giorni consecutivi), o ancora alle misure di sensori che si trovano in prossimità di fonti di illuminazione artificiale che vengono accese e spente. Queste caratteristiche rendono il segnale di luminosità un segnale irregolare, che quindi non può essere compresso oltre una certa soglia. Come si vede dalla Figura 6.2(c), la probagilità di trasmissione oscilla tra 1 e 0.8. Questo comportamento è facilmente giustificabile, infatti ogni volta che il modulo di ricostruzione esegue un'iterazione in cui tutti i campioni vengono raccolti, l'errore di ricostruzione risulta ovviamente nullo, quindi decide di abbassare la probabilità di trasmissione. All'iterazione successiva però, dopo aver ricostruito il segnale da un numero di campioni minore, la stima dell'errore



(a) Segnale di temperatura.



(b) Segnale di umidità.



(c) Segnale di luminosità.

Figura 6.2: Risultati Campagna **A**.

risulta superiore alla soglia impostata e quindi la p_{TX} viene riportata al valore iniziale.

Come si nota dal grafico, in alcuni casi la probabilità di trasmissione viene abbassata per 2 iterazioni successive. In questi casi isolati, anche se la p_{TX} risultava circa 0.9, è possibile che tutti i sensori abbiano trasmesso ugualmente, essendo la decisione presa in modo indipendente da ogni sensore. Il risultato è che la stima dell'errore risulta contenuta anche nell'iterazione in questione, la p_{TX} viene ulteriormente abbassata, finchè all'istante successivo l'algoritmo si accorge che l'errore sta aumentando e quindi alza nuovamente la probabilità di trasmissione.

Nella Tabella 6.2 sono riportati i valori medi $\overline{\xi}_R$ e $\overline{p}_{TX}$ per la Campagna **A**. Come si può vedere, l'errore quadratico medio è minore dell'1% sia per il segnale di temperatura, che per quello di umidità. Inoltre la probabilità di trasmissione media è di circa 20% per il primo segnale, e di circa il 25% per il secondo; ciò significa che durante ogni round della campagna di raccolta dati più del 70% dei sensori può fare a meno di trasmettere, con un gran risparmio di energia e aumento dell'efficienza. Il prezzo da pagare per questo risparmio di energia è l'errore commesso in ricostruzione, che però, per questi due segnali è comunque trascurabile. Per quanto riguarda il segnale di luminosità, esso viene ricostruito con un errore medio di poco più del 2%, ma per poter contenere questo errore, devono trasmettere in media il 95% circa dei sensori.

Segnale	$\overline{\xi}_R$	$\overline{p}_{TX}$
temperatura	0.0023	0.2045
umidità	0.0036	0.2552
luminosità	0.0209	0.9449

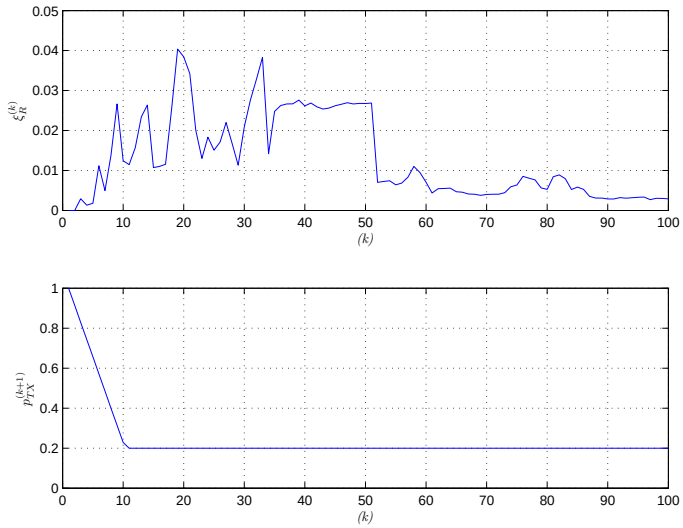
Tabella 6.2: Risultati medi per la campagna **A**.

6.2.2 Campagna B e Campagna C

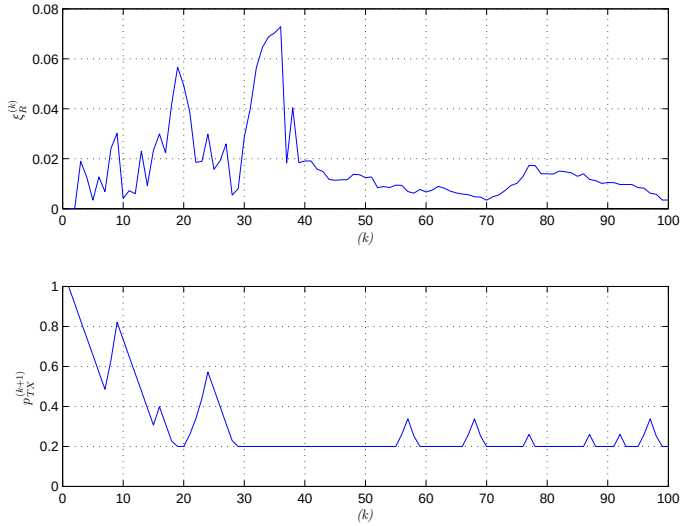
Le simulazioni effettuate su queste due campagne, riportati in Figura 6.3 per la campagna **B** ed in Figura 6.4 per la campagna **C**, confermano i risultati ottenuti. Si osservano gli stessi comportamenti da parte di tutti i segnali considerati, con l'eccezione del segnale di umidità della campagna **B**, Figura **B**(b), per cui risulta una $\overline{p}_{TX}$ di circa 73%, forse a causa di una maggior variabilità dei dati originali. L'errore medio comunque risulta ancora molto piccolo e questo dimostra la capacità del framework di adattarsi velocemente a segnali che si comportano in modo diverso. Nelle Tabelle 6.3 e 6.4 sono riportati i valori medi.

Segnale	$\overline{\xi}_R$	$\overline{p}_{TX}$
temperatura	0.0119	0.2049
umidità	0.0046	0.7290
luminosità	0.0281	0.9546

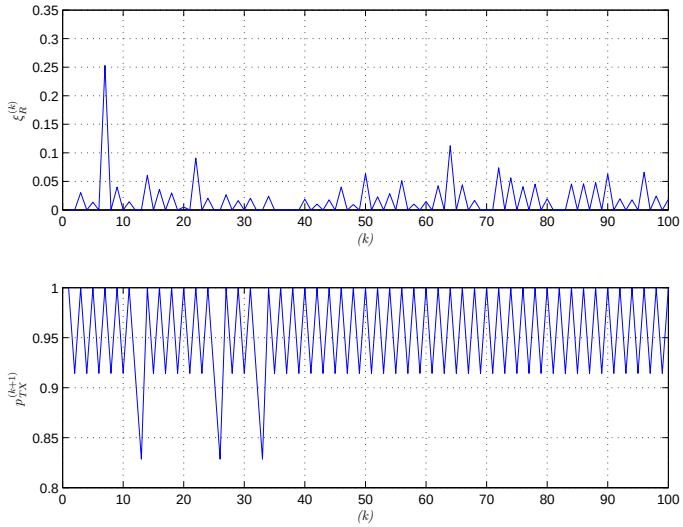
Tabella 6.3: Risultati medi per la campagna **B**.



(a) Segnale di temperatura.

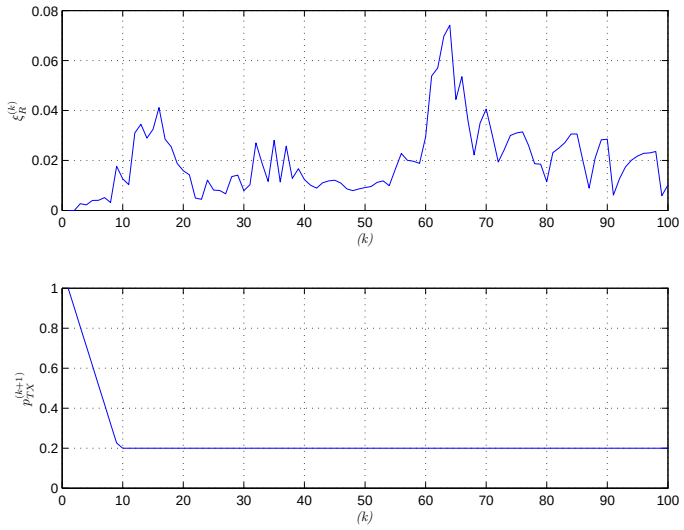


(b) Segnale di umidità.

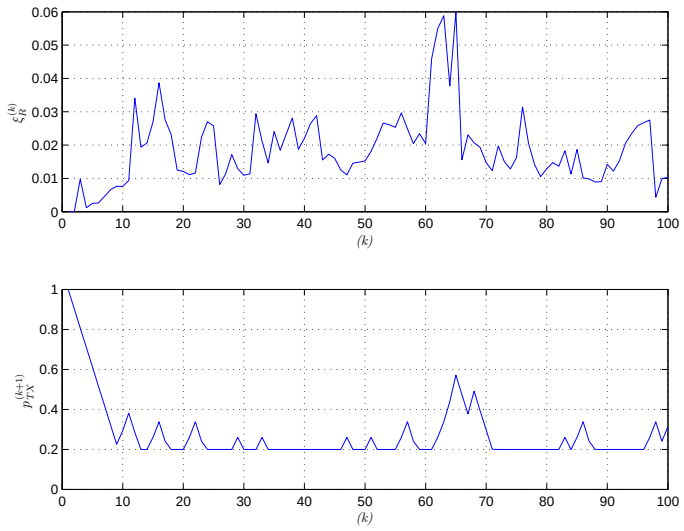


(c) Segnale di luminosità.

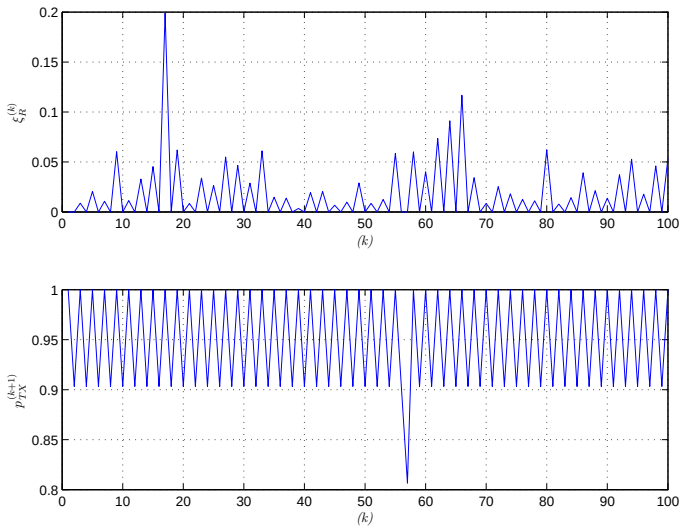
Figura 6.3: Risultati Campagna **B**.



(a) Segnale di temperatura.



(b) Segnale di umidità.



(c) Segnale di luminosità.

Figura 6.4: Risultati Campagna **C**.

Segnale	$\overline{\xi}_R$	$\overline{p}_{TX}$
temperatura	0.0096	0.2068
umidità	0.0094	0.3275
luminosità	0.0353	0.9504

Tabella 6.4: Risultati medi per la campagna **C**.

6.2.3 Campagna ABC

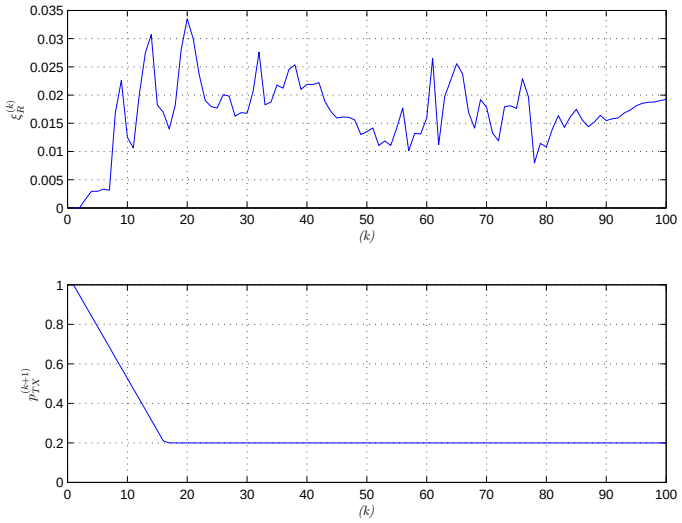
Unendo i dati delle campagne **A**, **B** e **C** è stata creata una campagna con un numero di sensori maggiore, così da poter testare il funzionamento dell'intero framework applicato a reti di circa un centinaio di sensori. In Figura 6.5 vengono riportati gli andamenti di $\xi_R^{(k)}$ e $p_{TX}^{(k+1)}$ per le prime 100 iterazioni. La scalabilità della tecnica Compressive Sensing è dimostrata dai risultati ottenuti anche in questo caso. I segnali di temperatura e umidità possono essere ricostruiti da pochi campioni, con un errore medio relativamente basso come riportato in Tabella 6.5, mentre il segnale di luminosità porta agli stessi risultati descritti in precedenza.

Segnale	$\overline{\xi}_R$	$\overline{p}_{TX}$
temperatura	0.0169	0.2648
umidità	0.0054	0.7025
luminosità	0.0143	0.9737

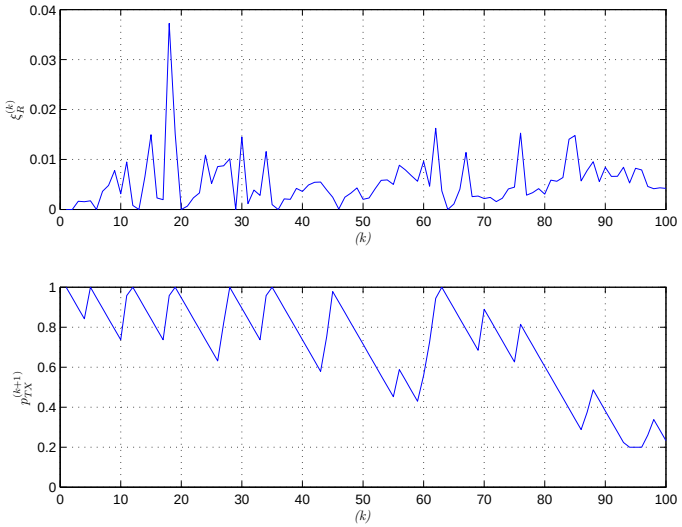
Tabella 6.5: Risultati medi per la campagna **ABC**.

6.2.4 Campagna LUCE

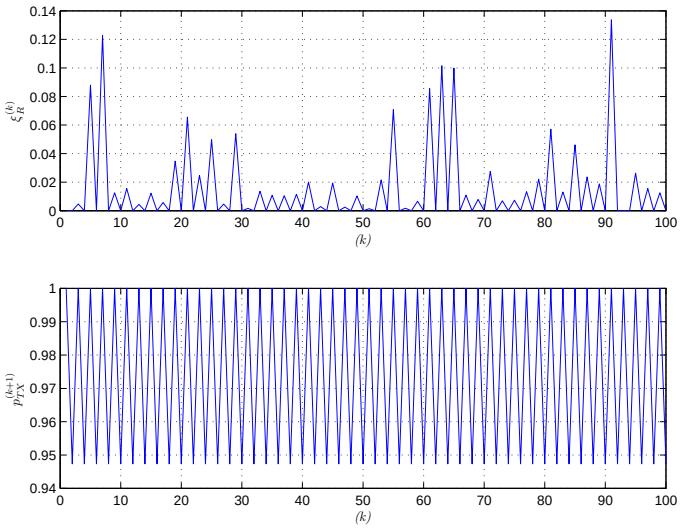
Il framework è stato ulteriormente testato con i segnali provenienti dalla rete LUCE [15]. I dati a disposizione riguardano, come riportato in Tabella 6.1, due misurazioni di temperatura, una al suolo e una ambientale, e una misura dell'umidità. La dimensione della rete è di 81 sensori e il numero di letture è 200; le letture sono state effettuate ad intervalli di 15 minuti. Gli andamenti di $\xi_R^{(k)}$ e $p_{TX}^{(k+1)}$ per i 3 tipi di segnale sono visualizzati in Figura 6.6, in Tabella 6.6 si riportano invece i risultati medi. Un intervallo di campionamento più grande, 15 minuti, si traduce in una diminuzione della correlazione temporale dei segnali in ingresso, e rende il recupero del segnale meno preciso. Di conseguenza le probabilità di trasmissione medie risultano elevate anche per segnali di temperatura, sopra il 70%, come riportato in Tabella 6.6. Il segnale di umidità invece non risulta affetto da questo problema, infatti il recupero avviene senza errore anche con una probabilità di trasmissione del 20%.



(a) Segnale di temperatura.

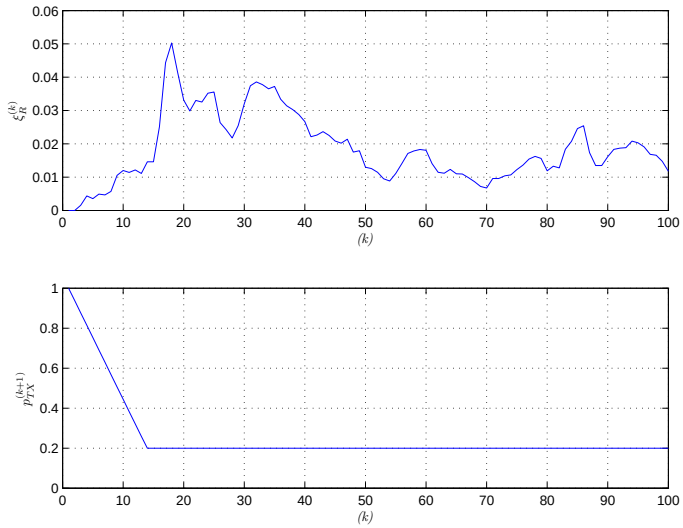


(b) Segnale di umidità.

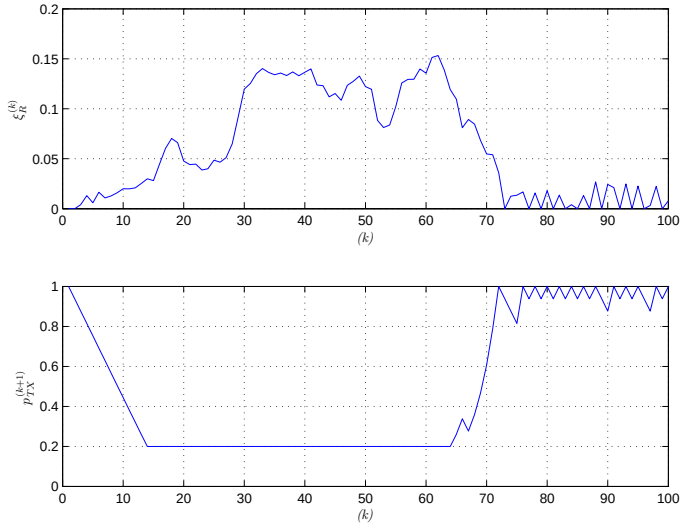


(c) Segnale di luminosità.

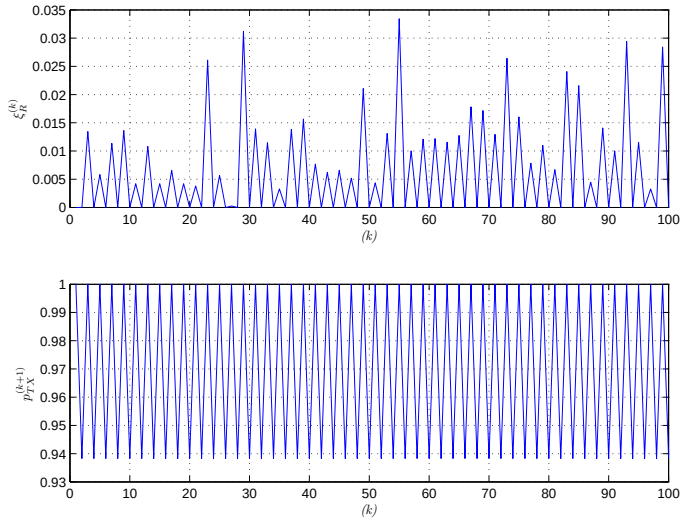
Figura 6.5: Risultati Campagna **ABC**.



(a) Segnale di umidità.



(b) Segnale di temperatura ambientale.



(c) Segnale di temperatura del suolo.

Figura 6.6: Risultati Campagna **LUCE**.

Segnale	$\overline{\xi}_R$	$\overline{p}_{TX}$
umidità	0.0140	0.2279
temp. ambiente	0.0357	0.7290
temp. suolo	0.0074	0.9688

Tabella 6.6: Risultati medi per la campagna **LUCE**.

6.2.5 Riassunto risultati

I risultati medi ottenuti nelle diverse simulazioni sono riportati in Tabella 6.7. In Figura 6.7 si rappresenta per le campagne **A**, **B**, **C** e **ABC** il valore medio ottenuto per l'errore di ricostruzione $\overline{\xi}_R$ (sull'asse delle ordinate) in funzione della probabilità media di trasmissione $\overline{p}_{TX}$ (sull'asse delle ascisse).

Come si può vedere nel grafico, i 12 punti si possono raggruppare in tre diverse "aree", individuate con delle ellissi, in corrispondenza delle quali il framework si comporta in modi diversi. Per le simulazioni che cadono contrassegnata dall'ellisse rossa, cioè quelle che riguardano i segnali di luminosità di ogni campagna, il framework raccoglie i dati con una probabilità media molto alta, tuttavia il corrispondente errore di ricostruzione risulta elevato. Per questa classe di segnali la tecnica del Compressive Sensing non porta vantaggi significativi se si vuole tenere la soglia di errore molto bassa, sotto il 5%, in quanto, con quasi lo stesso sforzo da parte della rete si possono collezionare i dati di tutti i sensori, e avere così a disposizione l'intero segnale evitando quindi l'errore di ricostruzione.

La seconda area, quella cerchiata in verde, raggruppa le simulazioni per cui la probabilità di trasmissione media è risultata circa 70%. Come si vede, applicare il Compressive Sensing in questo caso porta ad un aumento di efficienza della rete,

Campagna	Segnali	$\overline{\xi}_R$	$\overline{p}_{TX}$
A	temperatura	0.0023	0.2045
	umidità	0.0036	0.2552
	luminosità	0.0209	0.9449
B	temperatura	0.0119	0.2049
	umidità	0.0046	0.7290
	luminosità	0.0281	0.9546
C	temperatura	0.0096	0.2068
	umidità	0.0094	0.3275
	luminosità	0.0353	0.9504
ABC	temperatura	0.0169	0.2648
	umidità	0.0054	0.7025
	luminosità	0.0143	0.9737
LUCE	umidità	0.0140	0.2279
	temp. suolo	0.0074	0.9688
	temp. ambiente	0.0357	0.7290

Tabella 6.7: Riepilogo dei risultati medi ottenuti.

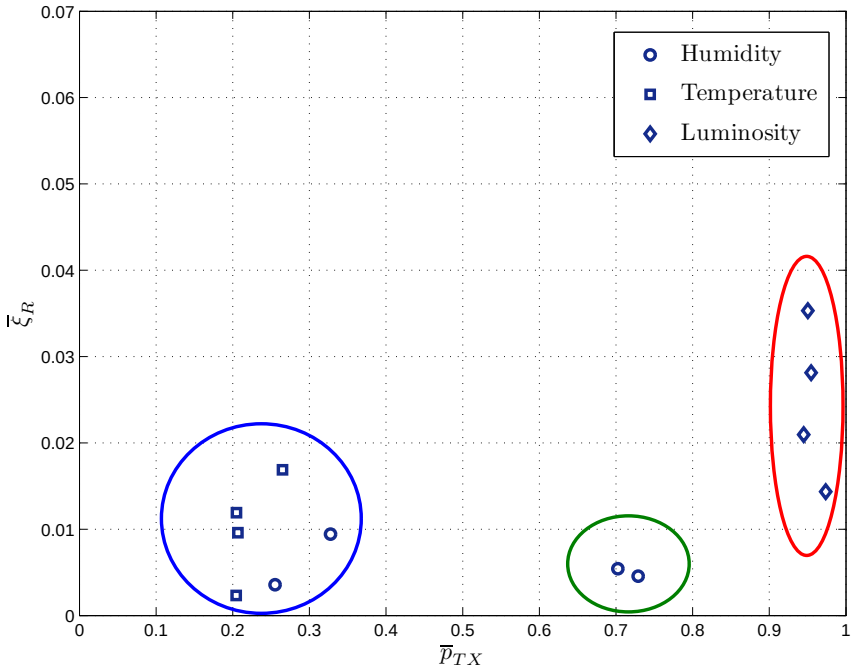


Figura 6.7: Visualizzazione dei risultati medi.

al costo di un errore di ricostruzione contenuto ($\overline{\xi}_R < 1\%$). Tenendo conto che la ricostruzione avviene all'esterno della rete, e che non comporta alcuna operazione aggiuntiva da parte dei sensori, ma solo un'ottimizzazione effettuata via software, si può concludere che questo è un buon risultato. Mediamente infatti circa un terzo dei sensori risparmia le proprie risorse energetiche ad ogni round di lettura.

Nell'ultima zona, contrassegnata nel grafico con l'ellisse blu, ricadono le simulazioni per cui il framework offre i risultati più significativi. Per questi segnali si ha un'ottimo aumento dell'efficienza della rete e l'errore di ricostruzione risulta comunque piccolo. Usando solo poco più del 20% delle risorse della rete si può infatti ottenere l'intero segnale con un errore minore del 2%.

6.3 Simulazioni con soglia variabile

Per capire come la soglia sulla stima dell'errore τ influenzi il comportamento del framework, sono state effettuate alcune simulazioni variando questo parametro. La soglia τ infatti è il riferimento per l'algoritmo di aggiornamento della probabilità di trasmissione. Una soglia alta comporta una maggiore tolleranza sull'errore di ricostruzione unitamente a una probabilità di trasmissione media maggiore, mentre impostando una soglia relativamente piccola, l'errore di ricostruzione viene abbassato al costo di una probabilità di trasmissione mediamente più alta.

Le simulazioni effettuate riguardano le campagne **A** e **LUCE**, scelti come campioni significativi dei segnali fin qui utilizzati. I vari parametri dell'algoritmo di ricostruzione sono stati mantenuti ai valori impostati nelle simulazioni prece-

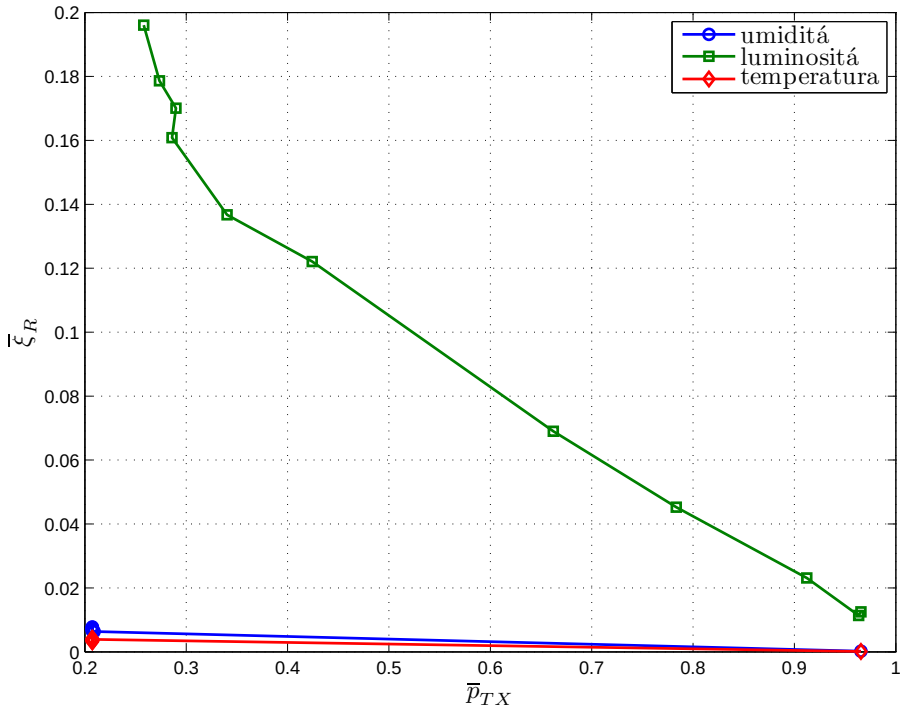


Figura 6.8: Prestazioni del framework per i segnali della campagna **A**.

denti, mentre la soglia τ è stata fatta variare da 0 a 0.4 a passi costanti di 0.04. L'obiettivo è quello di capire come l'errore di ricostruzione medio sia legato alla probabilità di trasmissione media, e visualizzare l'andamento delle prestazioni dell'algoritmo per ogni segnale considerato.

La Figura 6.8 mostra l'andamento di $\overline{\xi}_R$ in funzione di $\overline{p}_{TX}$ per la campagna **A**. Si può vedere come per i segnali di temperatura è umidità si ottengano delle prestazioni elevate, cioè un errore di ricostruzione molto piccolo anche con probabilità di trasmissione basse, mentre per il segnale di luminosità l'errore cresca abbastanza rapidamente quando la probabilità di trasmissione inizia a scendere. In particolare si nota come già con una p_{TX} media di circa l'80% l'errore superi il 5%, arrivando al 20% quando si tenta di recuperare il segnale da meno del 30% dei campioni, che per alcuni tipi di applicazioni può essere ancora considerato un risultato accettabile.

La Figura 6.9 riporta invece i risultati ottenuti per i segnali della campagna **LUCE**. In questo caso si può vedere come si riescano ad ottenere buone prestazioni per il segnale di umidità, mentre per i segnali di temperatura con probabilità di trasmissione minori di 0.6 l'errore non sia più trascurabile. Se però l'applicazione consente una tolleranza all'errore più elevata, attorno al 20%, anche per questi segnali è possibile fissare un valore di $\tau \approx 0.4$ che permette di recuperare il segnale di interesse trasmettendo circa il 30% dei dati e mantenendo allo stesso tempo l'errore di ricostruzione entro la soglia prestabilita.

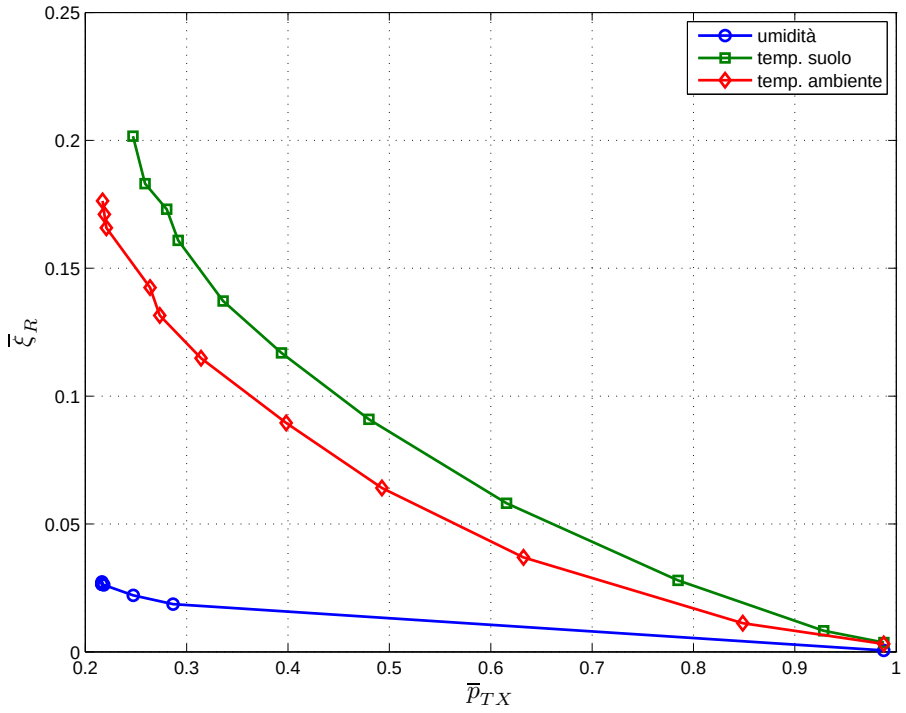


Figura 6.9: Prestazioni del framework per i segnali della campagna **LUCE**.

6.4 Osservazioni sull'applicazione del framework

Va fatto notare che il framework è implementato in modo che il monitoraggio della rete interessi un solo segnale alla volta, così da poter sfruttare al massimo le possibilità che Compressive Sensing offre per un dato segnale. In uno scenario in cui, ad ogni round, tutte le diverse letture di cui sono capaci i sensori vengano trasmesse alla Web application infatti, le prestazioni di Compressive Sensing sarebbero vincolate a quelle del "caso peggiore". Con riferimento ai segnali usati nelle simulazioni, se i sensori comunicano le letture di temperatura, umidità e luminosità nello stesso pacchetto, dopo una fase iniziale di qualche round, la probabilità di trasmissione si andrà a stabilizzare ad un valore alto, quello riferito al segnale di luminosità. Tutto il margine di miglioramento in termini di probabilità di trasmissione per i segnali di temperatura e umidità viene quindi a essere sprecato. Se si sta applicando la tecnica Compressive Sensing per preservare le risorse della rete è quindi necessario monitorare un segnale alla volta, oppure un insieme di segnali per cui le prestazioni siano simili, così da non ridurre troppo le performance.

7 Conclusioni

In questa tesi si è affrontato il problema della raccolta dati in una rete di sensori radio. Per aumentare l'efficienza della rete si è proposta una tecnica che permette la compressione dei dati che si vogliono collezionare e una successiva ricostruzione del segnale originale a partire da una sua versione incompleta. La ricostruzione viene effettuata al punto di raccolta e non comporta nessun aumento della complessità delle operazioni svolte dai sensori. La tecnica di compressione e ricostruzione proposta è il Compressive Sensing, affiancata alla Principal Component Analysis per trovare una matrice di trasformazione che renda sparso il segnale in ingresso e all'algoritmo di Nesterov per risolvere il problema di minimizzazione insito nel recupero del segnale. Oltre allo studio teorico, queste tecniche sono state implementate in un modulo $C++$, che è stato incorporato in un framework per il controllo e monitoraggio di reti di sensori radio.

Le prestazioni dell'algoritmo di Nesterov e dell'intero framework sono state valutate utilizzando segnali reali, provenienti da campagne di lettura precedentemente effettuate su due reti di sensori, e un software che simula il comportamento della WSN e del Gateway. Tale configurazione si è resa necessaria per poter avere a disposizione sia i dati ricostruiti che i dati originali e calcolare cosi l'errore di ricostruzione, ma non è la configurazione per cui è stato realizzato. Il framework è stato infatti implementato con lo scopo di applicare Compressive Sensing in tempo reale durante il monitoraggio di una rete di sensori. Per l'utilizzo in questa configurazione, è necessario realizzare lo strato applicativo da eseguire sul Gateway e il software adeguato per i sensori. Questi devono ovviamente rispettare lo stesso protocollo di comunicazione usato dalla Web application per permettere la raccolta dei dati e l'aggiornamento della probabilità di trasmissione.

I risultati ottenuti dalle diverse simulazioni mettono in luce le possibilità e i limiti di questa tecnica. L'interpretazione dei risultati dipende dalla tolleranza all'errore commesso in fase di ricostruzione che ha l'applicazione per cui si vogliono utilizzare i dati provenienti dai sensori. Per applicazioni in cui questa tolleranza è bassa ($< 5\%$) si nota come questa tecnica porta ad un notevole aumento dell'efficienza della rete quando applicato ad una certa classe di segnali da monitorare, mentre per altri le prestazioni non subiscono miglioramenti. Come illustrato nel Capitolo 6, i segnali che presentano una alta correlazione sia spaziale che temporale sono i candidati migliori per il Compressive Sensing, che riesce a mostrare

tutte le sue potenzialità se applicato con un giusto dimensionamento di tutti i parametri in gioco. Per questi segnali si ha infatti un notevole risparmio ($> 70\%$) in termini di trasmissioni, mantenendo l'errore sotto la soglia. Per segnali molto variabili invece non si riescono ad ottenere prestazioni accettabili, cioè l'errore di ricostruzione supera la soglia stabilita.

Se invece la tolleranza sull'errore è più elevata ($\sim 20\%$) i risultati mostrano come anche per segnali a bassa correlazione sia possibile ridurre il numero di trasmissioni, e quindi il consumo di energia, pur mantenendo l'errore sotto la soglia considerata.

Sviluppi Futuri

Un aspetto che può essere oggetto di ulteriore sviluppo è lo studio di qualche diversa tecnica per la costruzione delle matrici di trasformazione, che in questa tesi è svolta dalla Principal Component Analysis, ma che è possibile in numerosi modi diversi. Una matrice di trasformazione diversa si traduce in un diversa rappresentazione sparsa del segnale, fatto che può fare la differenza nell'errore commesso nel recupero del segnale. Un altro aspetto è lo studio di un'altra tecnica per la stima dell'errore di ricostruzione. Essendo d'obbligo l'uso di una tecnica euristica, potrebbero esserci differenze nella stima dell'errore. Va comunque detto che quella utilizzata in questa tesi si è dimostrata un'ottima soluzione che offre un buon compromesso tra qualità della stima prodotta e velocità di reazione a casi particolari. Infine si rende necessaria l'implementazione di un pannello di controllo nel framework WSNControl per dare all'utente la possibilità di regolare tutti i parametri utili ad applicare la tecnica Compressive Sensing in modo ottimale a seconda dello scenario.

Appendice A

Nel ricavare le espressioni (3.57) e (3.66) è stata usata la seguente identità:

$$\left(I + \frac{\lambda}{L_\mu} A^* A\right)^{-1} = \left(I - \frac{\lambda}{\lambda + L_\mu} A^* A\right) \ ,$$

con A^*A proiettore ortogonale. Per dimostrare tale risultato si ricordi che l'inversa si una matrice $P_{n\times n}$, se esiste, è una matrice $Q_{n\times n}$ tale per cui:

$$PQ = QP = I_n \ ,$$

dove I_n indica la matrice identità.

Nel caso in esame si ha:

-

$$\begin{aligned} PQ = \\ \left(I + \frac{\lambda}{L_\mu} A^* A\right)\left(I - \frac{\lambda}{\lambda + L_\mu} A^* A\right) = \\ I - \frac{\lambda}{\lambda + L_\mu} A^* A + \frac{\lambda}{L_\mu} A^* A - \frac{\lambda^2}{L_\mu(\lambda + L_\mu)} A^* \underbrace{AA^*}_{I} A = \\ I - \frac{\lambda}{\lambda + L_\mu} A^* A + \frac{\lambda}{L_\mu} A^* A - \frac{\lambda^2}{L_\mu(\lambda + L_\mu)} A^* A = \\ I - \frac{1}{\lambda + L_\mu}\left(\underbrace{-\lambda A^* A + \frac{\lambda^2}{L_\mu} A^* A + \lambda A^* A - \frac{\lambda^2}{L_\mu} A^* A}_{0}\right) = I \ . \end{aligned}$$

- $$\begin{aligned} QP = \\ \left(I - \frac{\lambda}{\lambda + L_\mu} A^* A\right)\left(I + \frac{\lambda}{L_\mu} A^* A\right) = \\ I - \frac{\lambda}{\lambda + L_\mu} A^* A + \frac{\lambda}{L_\mu} A^* A - \frac{\lambda^2}{L_\mu(\lambda + L_\mu)} A^* \underbrace{A A^*}_{I} A = \\ I - \frac{\lambda}{\lambda + L_\mu} A^* A + \frac{\lambda}{L_\mu} A^* A - \frac{\lambda^2}{L_\mu(\lambda + L_\mu)} A^* A = \\ I - \frac{1}{\lambda + L_\mu}\left(\underbrace{-\lambda A^* A + \frac{\lambda^2}{L_\mu} A^* A + \lambda A^* A - \frac{\lambda^2}{L_\mu} A^* A}_{0}\right) = I \ . \end{aligned}$$

Appendice B

Passaggi matematici che permettono di ricavare l'equazione (3.58):

$$\lambda \left(\|b - Ay_k\|_{\ell_2}^2 - \epsilon^2 \right) = 0 \, ,$$

$$\lambda \left(\left\| b - A \left(I - \frac{\lambda}{\lambda + L_\mu} A^* A \right) \left(\frac{\lambda}{L_\mu} A^* b + x_k - \frac{1}{L_\mu} \nabla f_\mu(x_k) \right) \right\|_{\ell_2}^2 - \epsilon^2 \right) = 0 \, ,$$

$$\lambda \left(\left\| b - \left(A - \frac{\lambda}{\lambda + L_\mu} \underbrace{AA^*}_{I} A \right) \left(\frac{\lambda}{L_\mu} A^* b + x_k - \frac{1}{L_\mu} \nabla f_\mu(x_k) \right) \right\|_{\ell_2}^2 - \epsilon^2 \right) = 0 \, ,$$

$$\lambda \left(\left\| b - A \left(\frac{L_\mu}{\lambda + L_\mu} \right) \left(\frac{\lambda}{L_\mu} A^* b + x_k - \frac{1}{L_\mu} \nabla f_\mu(x_k) \right) \right\|_{\ell_2}^2 - \epsilon^2 \right) = 0 \, ,$$

$$\lambda \left(\left\| \left(\frac{L_\mu}{\lambda + L_\mu} \right) \left(b - A \left(x_k - \frac{1}{L_\mu} \nabla f_\mu(x_k) \right) \right) \right\|_{\ell_2}^2 - \epsilon^2 \right) = 0 \, .$$

Le soluzioni possibili sono:

- nel caso il vincolo (3.54) non sia attivo $\lambda = 0$;
- nel caso il vincolo sia attivo invece:

$$\begin{aligned} \left\| \left(\frac{L_\mu}{\lambda + L_\mu} \right) \left(b - A \left(x_k - \frac{1}{L_\mu} \nabla f_\mu(x_k) \right) \right) \right\|_{\ell_2}^2 &= \epsilon^2 \, , \\ \left| \frac{L_\mu}{\lambda + L_\mu} \right| \left\| b - A \left(x_k - \frac{1}{L_\mu} \nabla f_\mu(x_k) \right) \right\|_{\ell_2} &= \epsilon \, , \\ L_\mu \left\| b - A \left(x_k - \frac{1}{L_\mu} \nabla f_\mu(x_k) \right) \right\|_{\ell_2} &= \epsilon(\lambda + L_\mu) \, . \end{aligned}$$

$$\lambda = \frac{L_\mu}{\epsilon} \left\| b - A \left(x_k - \frac{1}{L_\mu} \nabla f_\mu(x_k) \right) \right\|_{\ell_2} - 1 \, .$$

Passaggi matematici che permettono di ricavare l'equazione (3.67):

$$\lambda \left(\|b - Ay_k\|_{\ell_2}^2 - \epsilon^2 \right) = 0 \, ,$$

$$\lambda \left(\left\| b - A \left(I - \frac{\lambda}{\lambda + L_\mu} A^* A \right) \left(\frac{\lambda}{L_\mu} A^* b + x_k - \frac{1}{L_\mu} \sum_{i \leq k} \alpha_i \nabla f_\mu(x_i) \right) \right\|_{\ell_2}^2 - \epsilon^2 \right) = 0 \, ,$$

$$\lambda \left(\left\| b - \left(A - \frac{\lambda}{\lambda + L_\mu} \underbrace{AA^*}_{I} A \right) \left(\frac{\lambda}{L_\mu} A^* b + x_k - \frac{1}{L_\mu} \sum_{i \leq k} \alpha_i \nabla f_\mu(x_i) \right) \right\|_{\ell_2}^2 - \epsilon^2 \right) = 0 \, ,$$

$$\lambda \left(\left\| b - A \left(\frac{L_\mu}{\lambda + L_\mu} \right) \left(\frac{\lambda}{L_\mu} A^* b + x_k - \frac{1}{L_\mu} \sum_{i \leq k} \alpha_i \nabla f_\mu(x_i) \right) \right\|_{\ell_2}^2 - \epsilon^2 \right) = 0 \, ,$$

$$\lambda \left(\left\| \left(\frac{L_\mu}{\lambda + L_\mu} \right) \left(b - A \left(x_k - \frac{1}{L_\mu} \sum_{i \leq k} \alpha_i \nabla f_\mu(x_i) \right) \right) \right\|_{\ell_2}^2 - \epsilon^2 \right) = 0 \, .$$

Le soluzioni possibili sono:

- nel caso il vincolo (3.63) non sia attivo $\lambda = 0$;
- nel caso il vincolo sia attivo invece:

$$\begin{aligned} \left\| \left(\frac{L_\mu}{\lambda + L_\mu} \right) \left(b - A \left(x_k - \frac{1}{L_\mu} \sum_{i \leq k} \alpha_i \nabla f_\mu(x_i) \right) \right) \right\|_{\ell_2}^2 &= \epsilon^2 \, , \\ \left| \frac{L_\mu}{\lambda + L_\mu} \right| \left\| b - A \left(x_k - \frac{1}{L_\mu} \sum_{i \leq k} \alpha_i \nabla f_\mu(x_i) \right) \right\|_{\ell_2} &= \epsilon \, , \\ L_\mu \left\| b - A \left(x_k - \frac{1}{L_\mu} \sum_{i \leq k} \alpha_i \nabla f_\mu(x_i) \right) \right\|_{\ell_2} &= \epsilon(\lambda + L_\mu) \, . \end{aligned}$$

$$\lambda = \frac{L_\mu}{\epsilon} \left\| b - A \left(x_k - \frac{1}{L_\mu} \sum_{i \leq k} \alpha_i \nabla f_\mu(x_i) \right) \right\|_{\ell_2} - 1 \, .$$

Bibliografia

[1] R. Kay and F. Matter, "The design space of wireless sensor networks," *IEEE Wireless Communications*, vol. 11, no. 6, pp. 54–61, 2004.

[2] T. Haenselmann, *Sensornetworks*. GFDL Wireless Sensor Network textbook, 2006.

[3] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Comput. Networks*, vol. 38, 2002.

[4] E. J. Candès and M. B. Wakin, "An introduction to compressive sampling," *IEEE Signal Processing Magazine*, no. 21, 2008.

[5] R. Masiero, G. Quer, D. Munaretto, M. Rossi, J. Widmer, and M. Zorzi, "On the interplay between routing and signal representation for compressive sensing in wireless sensor networks," *Workshop on Information Theory and Applications*, vol. ITA, no. San Diego, 2009.

[6] M. A. T. Figueiredo, R. D. Nowak, and S. J. Wright, "Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems," *IEEE Journal of Selected Topics in Signal Processing*, vol. 1, pp. 586–597, 2007.

[7] E. T. Hale, W. Yin, and Z. Zhang, "Fixed–point continuation method for ℓ_1 regularized minimization with applications to compressed sensing," *Technical Report – Rice University*, 2007.

[8] ——, "Fixed–point continuation method for ℓ_1 minimization: Methodology and convergences," *SIAM J. on Optimization*, no. 19, pp. 1107–1130, 2008.

[9] K. Koh, S. J. Kim, and S. Boyd, "Solver fol ℓ_1 regularized least square problems," *Technical Report – Stanford University*, 2007.

[10] M. Visonà, "Sviluppo di una applicazione web per il monitoraggio di reti di sensori wireless," *Tesi di Laurea Specialistica – Università di Padova*, 2009.

[11] "Progetto europeo sensei." [Online]. Available: http://www.sensei-project.eu/

[12] J. W. Hui and D. E. Culler, "Extending ip to low-power, wireless personal area networks," *Internet Computing*, vol. 12, no. 4, pp. 37–45, 2008.

[13] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler, "Transmission of ipv6 packets over ieee 802.15.4 networks," *IETF RFC 4944*, 2007.

[14] "Piattaforma Java EE." [Online]. Available: http://java.sun.com/javaee/

[15] Ecole Polytechnique Fédérale de Lausanne, "LUCE (Lausanne Urban Canopy Experiment)." [Online]. Available: http://sensorscope.epfl.ch/index.php/LUCE

[16] D. Donoho, "Compressed sensing," *IEEE Trans. on Information*, vol. 52, no. 4, pp. 4036–4048, 2006.

[17] E. Candès, J. Romberg, and T. Tao, "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information," *IEEE Trans. on Information Theory*, vol. 52, no. 2, pp. 489–509, 2006.

[18] R. G. Baraniuk, M. Davenport, R. DeVore, and M. B. Wakin, "A simple proof of the restricted isometry principle for random matrices," *Constructive Approximation*, 2007.

[19] R. Masiero, G. Quer, D. Munaretto, M. Rossi, J. Widmer, and M. Zorzi, "Data acquisition through joint compressive sensing and principal component analysis," *IEEE GLOBECOM 2009*, vol. Honolulu Hawaii, 2009.

[20] M. Sciandrone, *Dispense sul metodo del gradiente*. Università di Firenze, 2008.

[21] A. Chiffi, *Analisi matematica vol. 2*. Padova: Alceo, 1998.

[22] A. Agnetis, *Introduzione all'ottimizzazione vincolata*. Università di Siena, 2008.

[23] Y. E. Nesterov, "A method for unconstrained convex minimization problem with rate of convergence $\wr(1/k^2)$," *Doklady AN USSR (Translate as Soviet Math. Docl.)*, p. 269, 1983.

[24] ——, *Introductory lectures on convex optimization: Basic course*. Boston: Kluwer, 2003.

[25] ——, "Smooth minimization of non–smooth functions," *Math. Program.*, vol. Serie A, no. 103, pp. 127–152, 2005.

[26] S. Becker, J. Bobin, and E. J. Candès, "Nesta: a fast and accurate first–order method for sparse recovery," *tech. report*, 2009.

[27] J. Bobin and E. J. Candès, "A fast and accurate first–order algorithm for compressed sensing," *submitted to ICIP09*, 2009.

[28] E. J. Candès, "ℓ_1-magic : Recovery of sparse signals via convex programming." [Online]. Available: http://www.acm.caltech.edu/l1magic/

[29] H. Mohimani, M. Babaie-Zadeh, and C. Jutten, "Smoothed l0 (sl0) algorithm for sparse decomposition." [Online]. Available: http://ee.sharif.edu/ SLzero/

[30] R. B. Davies, "C++ library NewMat10." [Online]. Available: http://www.robertnz.net/nm10.htm

Ringraziamenti

Giunto al termine di questo lavoro desidero ringraziare ed esprimere la mia riconoscenza nei confronti di tutte le persone che, in modi diversi, mi sono state vicine e hanno permesso e incoraggiato sia i miei studi che la realizzazione e stesura di questa tesi.

I miei più sentiti ringraziamenti vanno a chi mi ha seguito durante la redazione del lavoro di tesi: il Prof. Michele Rossi, per la fiducia fin da subito dimostratami nell'avermi proposto questo argomento di tesi; l'Ing. Giorgio Quer per avermi seguito durante lo svolgimento del lavoro con consigli e confronti che mi hanno aiutato ad intraprendere, ogni volta, le scelte più appropriate; l'Ing Riccardo Masiero per la rilettura critica dei capitoli matematici della tesi e per avermi guidato con i suoi suggerimenti durante lo studio degli argomenti teorici; l'Ing. Marco Visonà per la pazienza portata durante la fase di debugging del codice e quella simulativa. Senza il loro aiuto questo elaborato non sarebbe risultato della stessa qualità.

Desidero inoltre ringraziare la mia famiglia e gli amici che mi sono stati molto vicini in tutti questi anni, che oltre ad avermi sempre "supportato" mi hanno più di tutto "sopportato". Il mio primo pensiero, ovviamente, va ai miei genitori perchè senza il loro aiuto non avrei mai raggiunto questa meta. Sono davvero grato per tutto il sostengo economico, ma più di ogni altra cosa per quell'aiuto tacito o esplicito che è venuto dal loro cuore. Un ringraziamento speciale va a Sonia ♡, che con estrema pazienza ha sopportato i miei sbalzi di umore, le mie preoccupazioni e i momenti in cui i miei pensieri si concentravano su questa tesi lasciando poco spazio a tutto il resto. Se ho raggiunto questo traguardo lo devo anche alla sua continua presenza.

Come non ringraziare poi tutti gli amici dell'Università e in modo particolare i membri del CDSA con i quali ho condiviso più da vicino questi ultimi due anni di intenso studio (ma anche di piacevoli svaghi): Ponch, Agno, Neckerz, Leggenda, Ste, Cioa, Bruno, Marco, Leo, Fede, Ritto, Fiocco, Pippo, Manuel, David, Tobiz e Matteo. I miei coinquilini: Buz, Pape, Giorgio, Kevin, Matteo, Marco e Davide per tutte i momenti trascorsi insieme, per i tornei di PES e per l'amicizia fin da subito dimostratami. Gli amici storici: Andrea, Dalla, Edo, Federico, Christian, Rici e Stefano per i venerdì sera e le partite di calcetto che mi hanno dato modo di passare qualche momento di svago durante gli studi.

Rimarrà in me il piacevole ricordo di questi 5 (quasi 6) anni di studio trascorsi in questo dipartimento.

Aprile 2010,

Davide Zordan