Università di Padova Facoltà di Ingegneria

Dipartimento di Ingegneria dell'Informazione

Laurea Magistrale in Ingegneria delle Telecomunicazioni

Compressione spazio-temporale in reti di sensori tramite l'utilizzo combinato della Principal Component Analysis e della stima dei covariogrammi

Laureando

Emanuele Mentil

Relatore:
Prof. Michele Rossi

Correlatore:
Mohsen Hooshmand

Anno Accademico 2013/2014

A Chiara

SOMMARIO

Le reti di sensori radio stanno acquisendo sempre maggior importanza nell'ambito delle reti di telecomunicazioni. Esse sono formate da un insieme di sensori di modeste dimensioni e tipicamente alimentati a batteria, ognuno dei quali è equipaggiato con un microprocessore e diversi moduli di acquisizione e trasmissione dati. Grazie ai progressi fatti in ambito elettronico, i sensori ormai sono piccoli e posso essere posizionati a basso costo su diversi tipi di ambienti per monitorare nello spazio e nel tempo variazioni di grandezze fisiche quali la temperatura, l'umidità o la pressione. Per come vengono sviluppati oggi tali dispositivi, la trasmissione dati viene vista come il processo più dispendioso in termini energetici e quindi il design delle metodologie per raccogliere e trasmettere dati viene considerato come il problema centrale per questa tipologia di reti.

Un modo efficiente per vincere questa sfida è approssimare tramite modelli matematici e statistici l'evoluzione delle misure rilevate dai sensori nello spazio e/o nel tempo. Infatti, ogni qualvolta un modello teorico può essere usato in sostituzione delle misure reali, si possono ottenere guadagni significativi in termini di risparmio energetico nelle comunicazioni perché è possibile trasmettere un insieme di valori molto ristretto rispetto all'insieme di misure reali. Poiché nella maggior parte dei casi non ci sono informazioni a priori sulle misure prese dai sensori, il modello deve cercare di ottenerle in maniera automatica tramite l'utilizzo di tecniche di stima (parametrica e non), che permettono di modellare le variazioni delle misure future sulla base di quelle passate.

Il contributo di questa tesi è quello di studiare alcune strategie per la compressione

dei dati in reti di sensori radio utilizzando la tecnica del *Compressive Sensing* (CS) in combinazione con la *Principal Component Analysis* (PCA) per poi confrontarle con gli altri algoritmi proposti in letteratura. La peculiarità di questo approccio è l'abbattimento dei costi computazionali relativi alla compressione dei dati sul singolo nodo che è reso possibile grazie allo studio di alcune proprietà statistiche del segnale quali la sua matrice di covarianza. Questo permette di demandare tutto il carico di lavoro per la ricostruzione del segnale alla stazione radio base, garantendo così una maggior longevità alla rete.

INDICE

CAPITOLO 1

INTRODUZIONE

Con il termine WSN [1] (*Wireless Sensors Network*) si identifica una tipologia di rete composta da dispositivi elettronici autonomi in grado di comunicare con l'ambiente circostante e tra loro tramite una comunicazione radio. In particolare, il sensore è un dispositivo utile per misurare grandezze fisiche, come per esempio pressione, temperatura e luce, e grazie al loro costo ragionevolmente basso permette una raccolta di informazioni in tempo reale su aree sufficientemente vaste.

Purtroppo, in genere, il sensore ha ridotte capacità di calcolo, di memoria e di *throughput*, fa uso di una batteria (e di conseguenza ha risorse energetiche limitate), e solitamente viene posto in ambienti ostili e lontani da zone abitate. In particolare in un sensore si assume la trasmissione radio come il processo più dispendioso in termini energetici. Tutte queste problematiche fanno sì che si debba effettuare uno studio approfondito sul design della rete e sugli algoritmi processati dai sensori al fine di salvaguardare il più a lungo possibile la vita della WSN.

Lo scenario tipico per una rete di sensori prevedere un numero N di sensori sparso in una certa area, come per esempio un edificio o una foresta, al cui centro c'è un *sink* o stazione base dove vengono raccolti ed elaborati i dati prelevati dai sensori. Il *sink* non si preoccupa solo di elaborare i dati ricevuti, ma permette anche di collegare la rete al mondo esterno tramite Internet, rendendo la rete accessibile in qualsiasi momento e da qualsiasi parte del mondo. Ma mentre il *sink* in genere è una stazione fissa e collegata ad una rete di alimentazione, i sensori non lo sono ed essi mirano ad avere il massimo

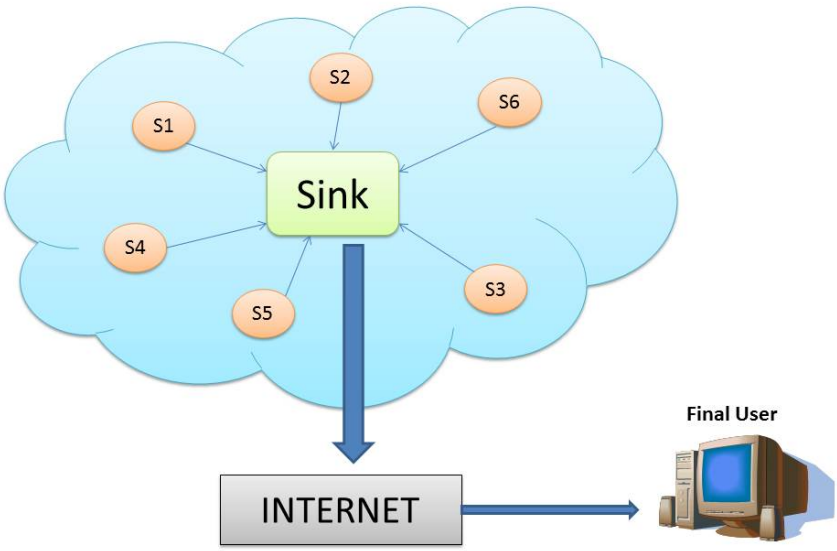


Figura 1.1. *Scenario tipico per una WSN*

risparmio energetico nella trasmissione di dati al *sink*.

Si prenda ora come esempio un vulcano attivo (Fig. 1.2). Esso è un luogo ostile per i sensori ed è solitamente lontano da centri abitati, quindi difficilmente raggiungibile dall'uomo. Si supponga che una rete di sensori sia stata installata tutt'attorno al vulcano ai fini di misurare microsismi e prevedere eventuali eruzioni che potrebbero danneggiare la popolazione locale. É ovvio pensare che il segnale misurato da ogni sensore sia grosso modo correlato con il segnale degli altri sensori presenti nella rete a causa della natura fisica del segnale stesso. Come nell'esempio, si ha che in genere i segnali reali in una certa area sono tra loro correlati e da questo ne segue la domanda: perché non sfruttare questa correlazione?

L'approccio usato in questa tesi è diminuire il consumo energetico sfruttando l'algoritmo *in-network* di compressione *Compressive Sensing* (CS) e la correlazione tra segnali. Il vantaggio principale di CS è che permette la ricostruzione (con ovviamente un certo errore) al *sink* dei segnali di N sensori usando solo l'informazione ricevuta da $L < N$ sensori. CS permette la trasmissione da parte di soli L sensori e quindi un incremento nel risparmio energetico. Per permettere a CS di funzionare in modo ottimale bisogna trovare una giusta base per la raffigurazione del sistema dei segnali, e

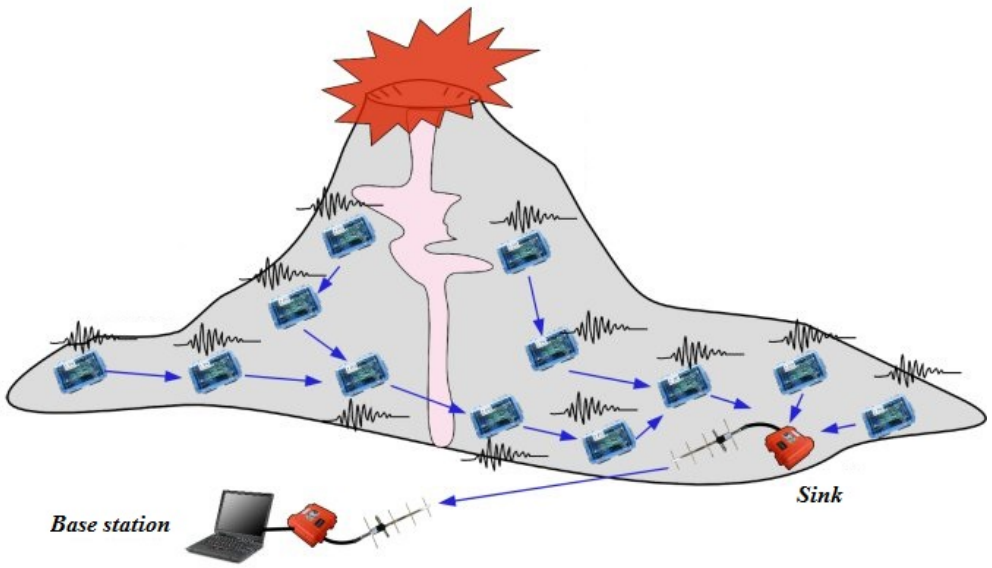


Figura 1.2. *Scenario: Vulcano attivo*

per far ciò si è usata la tecnica del *Principal Component Analysis* (PCA) che, tramite rotazioni del sistema di riferimento, permette di rappresentare al meglio un segnale N dimensionale usando solo $M < N$ variabili.

In aggiunta al PCA sono stati sviluppati degli algoritmi che si propongono di modellare in maniera ottima la matrice di correlazione dei segnali sulla base dei dati acquisiti precedentemente. Questi algoritmi sono l'argomento principale di questa tesi e gli sarà dedicato un intero capitolo più avanti.

Infine, per la simulazione in laboratorio, è stato utilizzato un generatore di segnali sintetici in cui è possibile variare la correlazione sia spaziale che temporale.

La struttura della tesi è così organizzata: nel secondo capitolo verranno illustrati gli strumenti di base utilizzati per la simulazione (CS e PCA), nel terzo capitolo verranno illustrati gli algoritmi di modellizzazione della matrice di correlazione, nel quarto verranno presentati i risultati ottenuti e infine nel quinto le conclusioni.

CAPITOLO 2

STRUMENTI UTILIZZATI

In questo capitolo si voglio presentare brevemente gli strumenti base utilizzati per la simulazione: le due tecniche di compressione e ricostruzione dati PCA e CS, il loro funzionamento congiunto, il concetto di funzione di correlazione e i segnali utilizzati per la simulazione.

2.1 PCA

Il Principal Component Analysis (PCA) [2] è uno strumento analitico per l'estrazione di informazioni rilevanti da un insieme di dati. Solitamente si esprime l'insieme di dati come combinazione **lineare** di una data base.

2.1.1 Introduzione al PCA

Prima di inoltrarci a vedere cos'è il PCA, definiamo $\mathbf{X}$ il nostro insieme di dati originali rappresentato come una matrice di dimensioni $N \times T$ dove N è il numero di osservanti (per esempio i sensori) e T è il numero di osservazioni temporali. L'assunzione di linearità fatta prima restringe il problema ad un semplice cambio di base della matrice $\mathbf{X}$ tramite una matrice di rotazione e stiramento $\mathbf{P}$ tale che la nuova matrice

$$\mathbf{Y} = \mathbf{PX}$$

rappresenti al meglio i segnali. Questo vuol dire massimizzare i segnali (massimizzare la loro varianza) e diminuire al minimo l'informazione mutua ridondante (minimizzare la covarianza tra essi).

Per capire come ottenere la miglior matrice $\mathbf{Y}$ si studi la sua matrice di covarianza, definita come

$$\mathbf{\Sigma_Y} = \frac{1}{T}\mathbf{Y}\mathbf{Y}^T \tag{2.1}$$

Si rappresenti ora $\mathbf{Y}$ come

$$\mathbf{Y} = \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \\ \dots \\ \mathbf{y}_N \end{bmatrix} \tag{2.2}$$

dove ogni vettore riga i-esima rappresenta la versione campionata del segnale registrato dal sensore i. Da come è definita la matrice di covarianza si ha che

$$\mathbf{\Sigma_{Y,(i,i)}} = \frac{1}{T}\mathbf{y}_i\mathbf{y}_i^T = \sigma_{i,i}^2 \tag{2.3}$$

ovvero la diagonale della matrice di covarianza rappresenta le varianze dei segnali mentre tutte le altre celle

$$\mathbf{\Sigma_{Y,(i,j)}} = \frac{1}{T}\mathbf{y}_i\mathbf{y}_j^T = \sigma_{i,j}^2 \tag{2.4}$$

rappresentano la covarianza tra il segnale i e il segnale j.

Considerando le proprietà desiderate per $\mathbf{Y}$ si ha che la matrice di covarianza $\mathbf{\Sigma_Y}$ deve avere tutti i valori non-diagonali nulli e la diagonale deve avere i valori ordinati in ordine di rango decrescente.

In pratica, il PCA cerca una matrice ortonormale $\mathbf{P}$ in modo che la matrice $\mathbf{\Sigma_Y} \equiv \frac{1}{T}\mathbf{Y}\mathbf{Y}^T$, dove $\mathbf{Y} = \mathbf{PX}$, sia una matrice diagonale. Le righe della matrice $\mathbf{P}$ vengono chiamate componenti principali (*principal components*) di $\mathbf{X}$.

Per trovare la matrice $\mathbf{P}$ si può procedere usando due metodi simili: quello degli autovettori e quello della *Singular Value Decomposition* (SVD).

2.1.2 Autovettori e autovalori

Prima di illustrare il procedimento degli autovalori e autovettori, si richiamano alcuni teoremi di algebra per una miglior comprensione del procedimento.

Theorem 2.1.1. *L'inversa di una matrice ortogonale è la sua trasposta:*

$$\mathbf{P}^{-1} = \mathbf{P}^T$$

Theorem 2.1.2. *Per una matrice* $\mathbf{A}$*,* $\mathbf{A}^T\mathbf{A}$ *e* $\mathbf{A}\mathbf{A}^T$ *sono simmetriche.*

$$(\mathbf{A}\mathbf{A}^T)^T = \mathbf{A}^{TT}\mathbf{A}^T = \mathbf{A}\mathbf{A}^T$$

$$(\mathbf{A}^T\mathbf{A})^T = \mathbf{A}^T\mathbf{A}^{TT} = \mathbf{A}^T\mathbf{A}$$

Theorem 2.1.3. *Una matrice è simmetrica se e solo se è ortogonalmente diagonalizzabile.*

Theorem 2.1.4. *Una matrice simmetrica è diagonalizzabile da una matrice di suoi autovettori ortonormali.*

Iniziando il ragionamento dalla matrice di covarianza di $\mathbf{Y}$, si ottiene

$$\begin{aligned}\mathbf{\Sigma_Y} &= \tfrac{1}{T}\mathbf{Y}\mathbf{Y}^T \\ &= \tfrac{1}{T}(\mathbf{P}\mathbf{X})(\mathbf{P}\mathbf{X})^T \\ &= \tfrac{1}{T}\mathbf{P}\mathbf{X}\mathbf{X}^T\mathbf{P}^T \\ &= \mathbf{P}\left(\tfrac{1}{T}\mathbf{X}\mathbf{X}^T\right)\mathbf{P}^T \\ &= \mathbf{P}\mathbf{\Sigma_X}\mathbf{P}^T\end{aligned}$$

dove

$$\mathbf{\Sigma_X} = \frac{1}{T}\mathbf{X}\mathbf{X}^T \tag{2.5}$$

è la matrice di covarianza della matrice $\mathbf{X}$.

Dal teorema 2.1.2, le matrici di covarianza $\mathbf{\Sigma_Y}$ e $\mathbf{\Sigma_X}$ sono delle matrici simmetriche $N \times N$. Grazie a questo e ai teoremi 2.1.3 e 2.1.4 si vuole procedere ora alla diagonalizzazione della matrice $\mathbf{\Sigma_Y}$ sfruttando il fatto che la matrice simmetrica $\mathbf{\Sigma_X}$ è diagonalizzabile in una matrice $\mathbf{D}$ tramite la matrice di autovettori di $\mathbf{\Sigma_X}$ ordinati in colonna seconda la relazione

$$\mathbf{\Sigma_X} = \mathbf{E}\mathbf{D}\mathbf{E}^T \tag{2.6}$$

dove $\mathbf{D}$ è una generica matrice diagonale e $\mathbf{E}$ è la matrice composta dagli autovettori di $\mathbf{\Sigma_X}$ ordinati in colonna.

Infine, scegliendo opportunamente $\mathbf{P} = \mathbf{E}^T$ e dal teorema 2.1.1 si ottiene

$$\begin{aligned} \mathbf{\Sigma_Y} &= \mathbf{P}\mathbf{\Sigma_X}\mathbf{P}^T \\ &= \mathbf{P}(\mathbf{E}^T\mathbf{D}\mathbf{E})\mathbf{P}^T \\ &= \mathbf{P}(\mathbf{P}^T\mathbf{D}\mathbf{P})\mathbf{P}^T \\ &= (\mathbf{P}\mathbf{P}^T)\mathbf{D}(\mathbf{P}\mathbf{P}^T) \\ &= (\mathbf{P}\mathbf{P}^{-1})\mathbf{D}(\mathbf{P}\mathbf{P}^{-1}) \\ &= \mathbf{D} \end{aligned} \tag{2.7}$$

Dalla 2.7 si evince che $\mathbf{\Sigma_Y}$ è la matrice diagonale di $\mathbf{\Sigma_X}$ ricavata usando i suoi autovettori raccolti nella matrice $\mathbf{P}$. In questo caso, essendo $\mathbf{P} = \mathbf{E}^T$, gli autovettori sono ordinati per righe.

In pratica, il calcolo del PCA si riduce ad un semplice calcolo di autovalori e autovettori della matrice di covarianza $\mathbf{\Sigma_X}$.

In altre parole si ottiene il seguente risultato. Si chiami $\mathbf{E}$ la matrice le cui colone sono gli autovettori calcolati da $\mathbf{\Sigma_X}$, disposti in modo tale che i relativi autovalori siano in ordine decrescente, e si definisca come $\bar{\mathbf{x}}$ la media temporale delle componenti del vettore $\mathbf{x}^k$ in una certa finestra (media in righe)

$$\bar{\mathbf{x}} = \frac{1}{K}\sum_{j=1}^{K}\mathbf{x}^{k-j} \tag{2.8}$$

Da questa relazione si può definire il vettore $\mathbf{s}^k$ come

$$\mathbf{s}^k = \mathbf{E}^T(\mathbf{x}^k - \bar{\mathbf{x}}) \tag{2.9}$$

Se il processo è correlato nel tempo, solo una porzione M di elementi di $\mathbf{s}^k$ è sufficiente per descrivere gli elementi di $\mathbf{x}^k - \bar{\mathbf{x}}$ e quindi si può descrivere il segnale $\mathbf{x}^k$ come

$$\mathbf{x}^k = \bar{\mathbf{x}} + \mathbf{E}\mathbf{s}^k \tag{2.10}$$

Il valore di M, e quindi la sparsità di $\mathbf{s}^k$ dipende da quanto è correlato il segnale.

2.1.3 SVD

Come per il metodo degli autovettori e autovalori, si vuole richiamare brevemente la teoria alla base del *Singular Value Decomposition* (SVD).

Sia $\mathbf{X}$ una matrice $N \times T$ arbitraria. La decomposizione a valori singolari della matrice $\mathbf{X}$ è defintia come

$$\mathbf{X} = \mathbf{UHV}^* \tag{2.11}$$

dove $\mathbf{U}$ e $\mathbf{V}$ sono matrici unitarie di dimensioni rispettivamente $N \times N$ e $T \times T$, $\mathbf{H}$ è una matrice di dimensioni $N \times T$ con valori positivi nella diagonale e nulli altrove e il * indica la coniugata trasposta. Gli elementi h_i di $\mathbf{H}$ sono detti *valori singolari* e corrispondono alle radici quadrate degli autovalori di $\mathbf{HH}^*$ o $\mathbf{H}^*\mathbf{H}$. Le N colonne di $\mathbf{U}$ sono dette *vettori singolari sinistri* e corrispondono agli autovettori di $\mathbf{HH}^*$, mentre le T colonne di $\mathbf{V}$ sono dette *vettori singolari destri* e corrispondono agli autovettori di $\mathbf{H}^*\mathbf{H}$.

Si richiama ora il seguente teorema e le seguenti definizioni.

Theorem 2.1.5. *Per ogni matrice* $\mathbf{X}$ *arbitraria di dimensioni* $m \times n$*, la matrice simmetrica* $\mathbf{X}^T\mathbf{X}$ *ha un insieme di autovettori ortonormali* $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n\}$ *e un insieme di autovalori associati* $\{\lambda_1, \lambda_2, \ldots, \lambda_n\}$*. L'insieme dei vettori* $\{\mathbf{Xv}_1, \mathbf{Xv}_2, \ldots, \mathbf{Xv}_n\}$ *forma una base ortogonale dove ogni vettore* $\mathbf{Xv}_i$ *è di lunghezza* $\sqrt{\lambda_i}$*.*

Si definisce $\mathbf{V}$ la matrice composta dagli autovettori ortonormali $\mathbf{v_i}$ disposti in colonna della matrice $\mathbf{X}^T\mathbf{X}$. Si definiscono λ_i gli autovalori di $\mathbf{X}^T\mathbf{X}$ associato agli autovettori $\mathbf{v_i}$. Si definiscono $h_i = \sqrt{\lambda_i} \in \mathbb{R}^+$ i valori singolari. Si definisce $\mathbf{U}$ la matrice composta dai vettori $\mathbf{u}_i$ disposti in colonna, dove $\mathbf{u}_i = \frac{1}{h_i}\mathbf{Xv}_i$.

Dalla definizione di autovettore si ha

$$(\mathbf{X}^T\mathbf{X})\mathbf{v}_i = \lambda_i\mathbf{v}_i \tag{2.12}$$

Dal teorema 2.1.5 si ricavano le seguenti proprietà:

- $\mathbf{u}_i \cdot \mathbf{u}_j = \begin{cases} 1 & i = j \\ 0 & \text{altrimenti} \end{cases}$;
- $|\mathbf{Xv}_i| = h_i$.

Da quanto detto fin'ora si può dimostrare il seguente risultato:

$$\mathbf{Xv}_i = h_i\mathbf{u}_i \tag{2.13}$$

Infatti moltiplicando a sinistra ambo i membri per $\mathbf{X}^T$ e sostituendo le definizioni si ha

$$\begin{aligned}\mathbf{X}^T\mathbf{X}\mathbf{v}_i &= h_i\mathbf{X}^T\mathbf{u}_i \\ &= h_i\mathbf{X}^T\frac{1}{h_i}\mathbf{X}\mathbf{v}_i \\ &= \mathbf{X}^T\mathbf{X}\mathbf{v}_i\end{aligned}$$

In una versione matriciale dove $\mathbf{H}$ è la matrice composta dai valori singolari nella diagonale e nulla altrove, si ottiene la seguente equazione

$$\mathbf{XV} = \mathbf{UH} \tag{2.14}$$

Data l'ortogonalità di $\mathbf{V}$ si ha che $\mathbf{V}^{-1} = \mathbf{V}^T$ e quindi

$$\mathbf{X} = \mathbf{UHV}^T \tag{2.15}$$

Ora, ritornando alla matrice originale $\mathbf{X}$ di dimensioni $N \times T$ contenente i segnale dei sensori e definendo una nuova matrice $T \times N$ come

$$\mathbf{Y} = \frac{1}{\sqrt{T}}\mathbf{X}^T \tag{2.16}$$

dove ogni colonna di $\mathbf{Y}$ ha media nulla, si ottiene che

$$\mathbf{Y}^T\mathbf{Y} = \left(\frac{1}{\sqrt{T}}\mathbf{X}^T\right)^T\left(\frac{1}{\sqrt{T}}\mathbf{X}^T\right) \tag{2.17}$$

$$= \frac{1}{T}\mathbf{X}\mathbf{X}^T \tag{2.18}$$

$$= \mathbf{\Sigma_X} \tag{2.19}$$

ovvero, per costruzione, $\mathbf{Y}^T\mathbf{Y}$ corrisponde alla matrice di covarianza di $\mathbf{X}$.

Considerando quanto detto alla fine della sottosezione 2.1.1, si nota che le componenti principali di $\mathbf{X}$ sono gli autovettori di $\mathbf{\Sigma_X}$. Se si calcola l'SVD di $\mathbf{Y}$ si può notare che le colonne della matrice $\mathbf{V}$ contengono gli autovettori di $\mathbf{Y}^T\mathbf{Y} = \mathbf{\Sigma_X}$. In altre parole, le colonne di $\mathbf{V}$ contengono le componenti principali di $\mathbf{X}$.

In pratica si ha che $\mathbf{V}$ descrive lo spazio delle righe di $\mathbf{Y} = \frac{1}{\sqrt{T}}\mathbf{X}^T$, e quindi anche lo spazio delle colonne di $\frac{1}{\sqrt{T}}\mathbf{X}$. Quindi, in conclusione, trovare le componenti principali equivale a trovare una base ortonormale che descrive lo spazio delle colonne di $\mathbf{X}$.

2.2 CS

Compressive Sensing [3] [14] è una tecnica che permette di recuperare un dato segnale N dimensionale usufruendo solo di L campioni con l'ipotesi che $L << N$. Come nella precedente sezione, si consideri il segnale X come una matrice di dimensioni $N \times T$ dove N è il numero di sensori e T il numero di istanti temporali considerati. Si indichi con $\mathbf{x}^k \in \mathbb{R}^N$ i valori prelevati dai sensori all'istante temporale k e lo raffiguriamo come un vettore colonna di dimensioni $N \times 1$.

Si definisca in aggiunta una matrice $\mathbf{\Psi}$ di dimensioni $N \times N$ tale che

$$\mathbf{x}^k = \mathbf{\Psi}\mathbf{s}^k \tag{2.20}$$

dove il vettore $\mathbf{s}^k \in \mathbb{R}^N$ è un vettore M-sparso, ovvero che ha solo M valori su N significativi rispetto all'energia medie per componente

$$E_{\mathbf{s}^k} = \frac{1}{N}\sqrt{\langle \mathbf{s}^k, \mathbf{s}^k \rangle} \tag{2.21}$$

Le altre $N - M$ componenti sono trascurabili.

Assumendo di conoscere $\mathbf{\Psi}$, si può ricavare $\mathbf{s}^k$ tramite

$$\mathbf{s}^k = \mathbf{\Psi}^{-1}\mathbf{x}^k$$

In aggiunta, $\mathbf{s}^k$, può essere ottenuto da un numero L di proiezioni casuali di $\mathbf{x}^k$ (chiamate $\mathbf{y}^k \in \mathbb{R}^L$), con le condizioni che valga $M \leqslant L < N$ e che valga

$$\mathbf{y}^k = \mathbf{\Phi}\mathbf{x}^k \tag{2.22}$$

dove M è il numero di componenti significative rispetto a 2.21, L è il numero di campioni disponibili e N è il numero totale di sensori.

Nel contesto di questa tesi, la matrice $\mathbf{\Phi}$ può essere vista come una matrice di *routing* con dimensioni $L \times N$, poiché è composta da soli zeri e uni, ed ogni sua riga ha uno ed uno solo 1, e ogni sua colonna ha al più un solo 1. La presenza di un 1 indica che il sensore associato a tale cella della matrice sta campionando il segnale.

Unendo quanto detto fin'ora si ha che

$$\begin{aligned} \mathbf{y}^k &= \mathbf{\Phi}\mathbf{x}^k \\ &= \mathbf{\Phi}\mathbf{\Psi}\mathbf{s}^k \\ &= \tilde{\mathbf{\Phi}}\mathbf{s}^k \end{aligned}$$

dove $\tilde{\mathbf{\Phi}}$ è una matrice $L \times N$ definita come

$$\tilde{\mathbf{\Phi}} = \mathbf{\Phi}\mathbf{\Psi}$$

In Fig. 2.1 si può vedere un esempio del prodotto matriciale di CS.

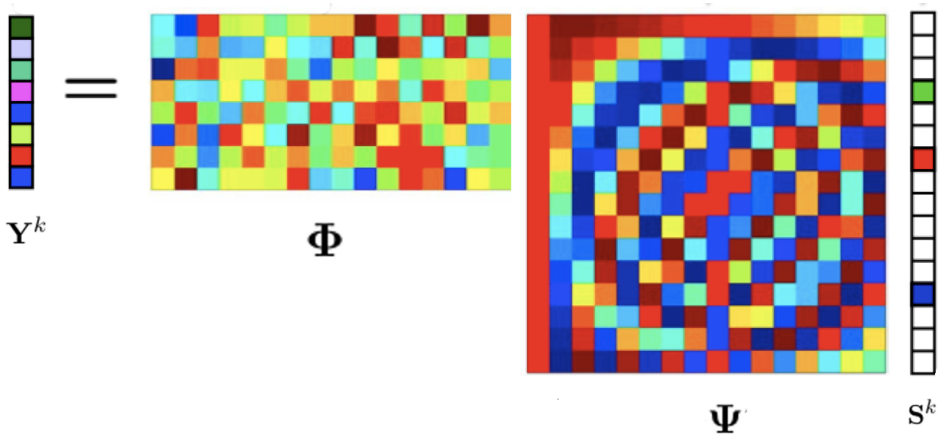


Figura 2.1. *Prodotto matriciale di CS*

Il sistema risultante è purtroppo mal condizionato poiché abbiamo L equazioni ed N variabili. Se il vettore $\mathbf{s}^k$ è sparso e il prodotto $\mathbf{\Phi}\mathbf{\Psi}$ soddisfa la condizione di RIP [12], si dimostra che è possibile invertire il sistema

$$\mathbf{y}^k = \tilde{\mathbf{\Phi}}\mathbf{s}^k$$

e quindi ricavarsi $\mathbf{s}^k$ e $\mathbf{x}^k$ (quest'ultimo grazie all'equazione 2.20).

2.3 PCA e CS

Fin'ora si è visto come il PCA permetta di rappresentare al meglio un segnale N dimensionale usando solo $M < N$ variabili e come CS sia una tecnica che permette di recuperare un segnale N dimensionale tramite la lettura di $L < N$ campioni. L'unione dei due metodi viene presentata di seguito.

Si assuma di essere all'istante k e si assuma di conoscere tutta l'informazione precedente a tale istante (per esempio si è salvata tutta l'informazione stimata al *sink* fino a tale istante o magari solo l'informazione riguardante i K precedenti istanti temporali).

Tramite PCA si può mappare il segnale $\mathbf{x}^k$ in un vettore sparso $\mathbf{s}^k$, ovvero, dall'insieme di dati salvati si può procedere al calcolo di

- $\mathbf{\Sigma^{k-1}}$, la matrice di covarianza;
- $\mathbf{E}^k$, la matrice di autovettori ricavata da $\mathbf{\Sigma^{k-1}}$;
- $\bar{\mathbf{x}}^k$, la media temporale del segnale tramite (2.8).

e dall'equazione (2.10) si ottiene

$$\mathbf{x}^k - \bar{\mathbf{x}}^k = \mathbf{E}^k \mathbf{s}^k \tag{2.23}$$

Di conseguenza tramite l'equazione (2.22) si ottiene

$$\begin{aligned} \mathbf{y}^k - \mathbf{\Phi}^k \bar{\mathbf{x}}^k &= \mathbf{\Phi}^k \mathbf{x}^k - \mathbf{\Phi}^k \bar{\mathbf{x}}^k \\ &= \mathbf{\Phi}^k (\mathbf{x}^k - \bar{\mathbf{x}}^k) \\ &= \mathbf{\Phi}^k \mathbf{E}^k \mathbf{s}^k \end{aligned}$$

Con il fatto che anche la matrice di *routing* $\mathbf{\Phi}^k$ è dipendente dall'istante temporale k si vuole indicare che essa cambia nel tempo a seconda di quali sensori si accendono e campionano il segnale. Ne consegue che trovando buone approssimazioni [12] di $\hat{\mathbf{s}}^k$ è possibile trovare una buona stima di $\hat{\mathbf{x}}^k$ tramite

$$\hat{\mathbf{x}}^k = \bar{\mathbf{x}}^k + \mathbf{E}^k \mathbf{s}^k \tag{2.24}$$

2.4 Funzione di correlazione

Lo scopo principale di questa tesi è quello di migliorare i risultati grezzi ottenuti con CS e PCA, migliorando la stima della matrice di covarianza $\mathbf{\Sigma}$ da passare a PCA. L'idea di base è quella di generare un modello dai dati precedentemente acquisiti e stimati e raffinare la stima della matrice di covarianza tramite quel modello.

Ciò che si è andati a modellizzare è la **funzione di correlazione**. La funzione di correlazione indica quanto sono correlati due segnali aventi una certa distanza spaziale h tra essi. Per il concetto di funzione di correlazione si è preso spunto dal concetto di **variogramma** del Kriging [4] [5] usato principalmente per creare modelli di segnali molto correlati spazialmente.

Non si è voluto utilizzare la covarianza poiché le ampiezze di covarianza tra i vari sensori non sono comparabili tra loro, mentre, essendo la correlazione normalizzata secondo le deviazioni standard, la comparazione è possibile.

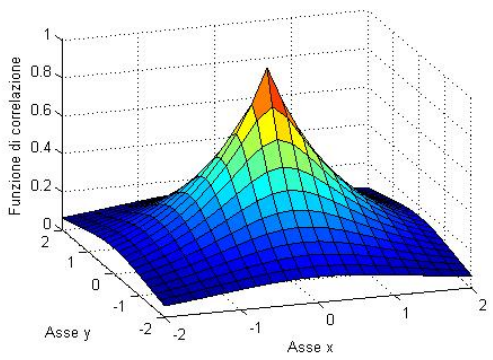


Figura 2.2. *Versione tridimensionale*

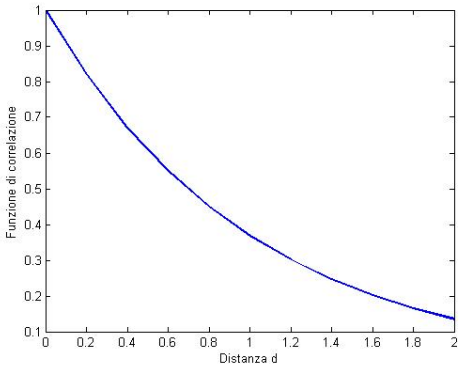


Figura 2.3. *Versione bidimensionale*

Come spesso accade nei variogrammi del Kriging in geostatistica [13], la funzione di correlazione viene assunta isotopica e quindi si è potuto passare da un modello tridimensionale ad un modello bidimensionale. In Fig. 2.2 e Fig. 2.3 ci sono due esempi di funzione di correlazione nella sua versione tridimensionale e bidimensionale in un modello esponenziale. Dato che la funzione di correlazione si ricava dalla matrice di correlazione, e questa a sua volta è stata ricavata dai valori temporali di tutti i nodi, essa ingloba l'informazione relativa sia alla correlazione temporale che alla correlazione spaziale.

Dato un insieme di segnali salvati in una matrice $\mathbf{X}$ di dimensioni $N \times T$ dove N è il numero di sensori e T è il numero di istanti temporali considerati, si calcola la loro matrice di covarianza e di conseguenza la relativa matrice di correlazione (il procedimento completo utilizzato è descritto nel prossimo capitolo). Dalla matrice di correlazione si passa a riordinare i suoi valori in funzione della distanza euclidea tra i sensori. A questo punto, ottenuta una versione grezza delle funzione di correlazione $f_P(d)$, si passa alla stima del modello che meglio adatta i valori trovati.

I modelli utilizzabili sono molteplici. I più conosciuti sono:

- il modello esponenziale

$$f_E(a, b, d) = ae^{bd} \tag{2.25}$$

- il modello gaussiano

$$f_G(a, b, d) = ae^{bd^2} \tag{2.26}$$

- il modello sferico

$$f_S(a,b,d) = a\left[1 - 1.5\left(\frac{d}{b}\right) + 0.5\left(\frac{d}{b}\right)^3\right] \tag{2.27}$$

ma nessuno vieta di utilizzarne altri. In Fig. 2.4 si possono vedere un esempio dei modelli sopra citati.

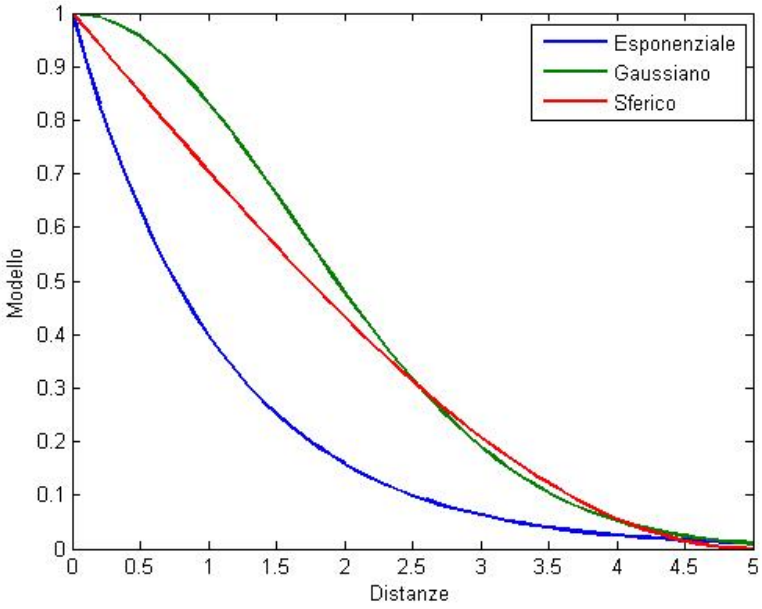


Figura 2.4. *Modelli d'esempio*

Trattandosi di correlazione, si può fare una piccola osservazione sul parametro a. Alla distanza $d = 0$, si sta stimando la correlazione che c'è tra un sensore e se stesso e la correlazione del sensore con se stesso non può essere altro che 1. Ne segue quindi che si può evitare di considerare il parametro a come una variabile da determinare nella funzione di fit, e considerarlo semplicemente unitario.

Possibili metriche per scegliere quale modello e quali parametri si adattano meglio alla funzione di correlazione grezza sono

- il *mean square error*

$$MSE(a,b,m) = \frac{1}{J}\sum_d \big(f_P(d) - f_m(a,b,d)\big)^2 \tag{2.28}$$

dove m indica il modello (E,G,S), a e b sono i parametri del modello scelto e J è il numero totale di tutte le possibili distanze d.

- la norma p

$$NP(a,b,m,p) = \sqrt[p]{\sum_{d} \left(f_P(d) - f_m(a,b,d)\right)^2} \tag{2.29}$$

  dove p indica la norma prescelta: con $p = 1$ si ha il valore assoluto, $p = 2$ si ha la norma euclidea, ...

Infine, una volta trovato il modello che meglio si adatta ai dati, si passa alla creazione delle matrice di covarianza sfruttando come valore di correlazione il valore restituito dal modello, considerando come distanza la distanza fra i nodi e come parametri quelli stimati in precedenza.

2.5 Segnali utilizzati

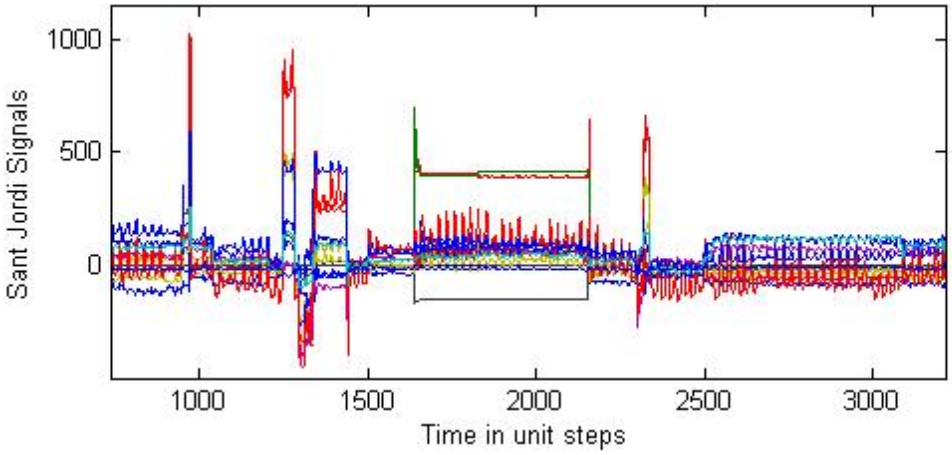


Figura 2.5. *Esempio di segnali del Sant Jordi*

Per la simulazione sono stati utilizzati due tipi di segnali:

- segnali reali, presi da circa 150 sensori che ogni 5 secondi monitoravano le tensioni strutturali dello stadio Sant Jordi di Barcellona in tutto il 2013;

- segnali sintetici generati da un generatore precedentemente implementato.

Il generatore di segnali sintetici [6] è stato particolarmente utile per via del fatto che permette un buon *tuning* del segnale, consentendo all'utilizzatore di regolare a piacere l'area del segnale, la correlazione spaziale e la correlazione temporale. Nelle figure Fig.

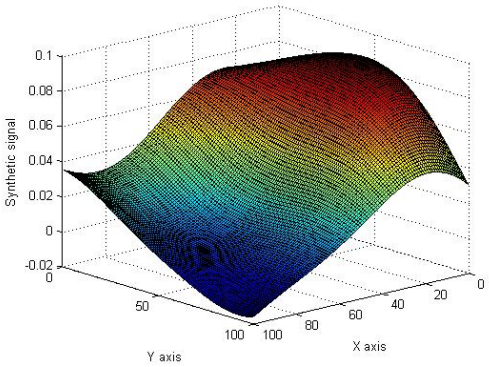


Figura 2.6. *Segnale molto correlato*

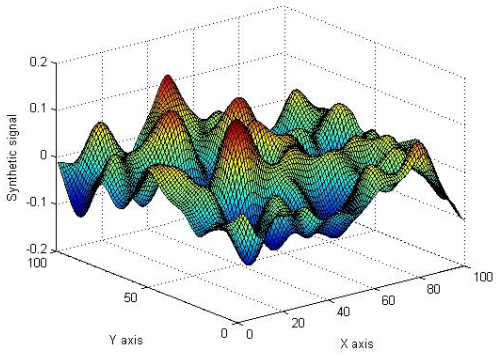


Figura 2.7. *Segnale poco correlato*

2.6 e Fig. 2.7 si possono vedere rispettivamente due esempi di alta correlazione spaziale e bassa correlazione spaziale su un'area di 100×100 unità.

CAPITOLO 3

FRAMEWORK

Do ora in avanti, per semplicità di notazione, quando si parla di matrice di covarianza $\mathbf{\Sigma}$ ci si riferisce alla matrice di covarianza $\mathbf{\Sigma_X}$. Inoltre, data una matrice $\mathbf{A}$ si denota con $\mathbf{A}^k$ la matrice all'istante temporale k e con $\mathbf{A}_{i,j}$ il valore alla riga i e alla colonna j della matrice.

In questo capitolo si andrà ad analizzare due algoritmi, i quali compongono il lavoro principale di questa tesi. L'obiettivo di tali algoritmi è la modellizzazione della matrice di correlazione al fine di ottenere risultati migliori con PCA-CS.

L'idea base di entrambi gli algoritmi è quella di trovare un modello per stimare la matrice di covarianza $\mathbf{\Sigma}$ dalla quale, tramite CS, si stima i possibili valori del segnale originale $\hat{\mathbf{x}}$ all'istante di tempo corrente. In pratica, assumendo di essere all'istante k, si procede nel seguente modo:

1. si calcola la matrice di covarianza $\mathbf{\Sigma}^{k-1}$ basata sui valori di $\hat{\mathbf{x}}$ stimati precedentemente;

2. si calcola da $\mathbf{\Sigma}^{k-1}$ la matrice di correlazione $\mathbf{P}$ in modo tale che le ampiezze dei valori di tale matrice siano comparabili tra loro;

3. si ordinano tutti i valori della matrice di correlazione in un vettore in base alla distanza fra i vari nodi e si calcola la funzione di correlazione;

4. tramite questa si stimano i parametri del nostro modello di tale funzione;

5. infine si crea la matrice $\mathbf{\Sigma}^k$ tramite i parametri appena trovati e da qui si procede alla stima di $\hat{\mathbf{x}}^k$ tramite PCA-CS.

Prima di inoltrarsi nelle funzioni utilizzate per le simulazioni è importante notare che gli indici sono in notazione Matlab quindi partono da 1 e non da 0 come in C/C++.

3.1 CovarianceEstimation (VarEst)

Assumendo di essere all'istante temporale k, il primo metodo di stima semplicemente calcola la matrice di covarianza $\mathbf{\Sigma}^{k-1}$ usando tutta l'informazione disponibile, ovvero usando tutti i valori di $\hat{\mathbf{x}}$ stimati fino all'istante $k-1$. Trovata la matrice di covarianza, il metodo calcola la matrice di correlazione $\mathbf{P}$ tramite la formula

$$\mathbf{P}_{i,j} = \frac{\mathbf{\Sigma}_{i,j}^{k-1}}{\sigma_i \sigma_j} \tag{3.1}$$

dove

$$\sigma_i = \sqrt{\mathbf{\Sigma}_{i,i}^{k-1}}$$

è la deviazione standard del nodo i.

Questo permette una comparazione tra i vari valori della matrice al variare dei nodi (identificati dalle righe e dalle colonne della matrice) perché a questo punto i valori di covarianza sono stati normalizzati tramite le rispettive deviazioni standard. Tramite la matrice di correlazione, e conoscendo le distanze fra i vari nodi, si può ricostruire la funzione di correlazione definita come

$$f_{\mathbf{P}}(h) = E[\mathbf{P}_{i,j}, \quad \forall\, (i,j) : d(i,j) = h] \tag{3.2}$$

dove con $E[]$ si intende la funzione di media e con $d(i,j)$ si intende la distanza Euclidea tra il nodo i e il nodo j:

$$d(i,j) = \sqrt{(i_x - j_x)^2 + (i_y - j_y)^2} \tag{3.3}$$

con i_x e j_x sono le coordinate x del nodi i e j e i_y e j_y sono le coordinate y sempre del nodo i e j.

Lo scalare h è ovviamente un valore reale e, considerando che la posizione dei nodi è assunta casuale, ci si può ritrovare anche N distanze distinte e, se N è un numero

elevato, la cosa potrebbe risultare problematica per un calcolatore. Per esempio, le distanze $13,3455m$, $13,346m$ e $13,3465m$ sono distinte tra loro, ma a livelli pratici uno spostamento di mezzo millimetro è privo di senso e quindi a livello di simulazione risulta più comodo quantizzare le distanze tra i vari nodi. Chiamando H un insieme di distanze quantizzate e $h_Q \in H$ un valore di tale insieme si ridefinisce così la funzione di correlazione come

$$f_{\mathbf{P}}(h_Q) = E\left[\mathbf{P}_{i,j}, \quad \forall\,(i,j) : \underset{h_Q}{\operatorname{argmin}}\big\{d(i,j) - h_Q\big\}\right] \tag{3.4}$$

Una volta calcolati i valori della funzione di correlazione, si passa alla creazione del modello che adatta in maniera ottima tali valori. I metodi di fit sono meglio illustrati nella sezione 3.3. Il modello considerato è quello esponenziale ovvero

$$\rho(h) = ae^{bh} \tag{3.5}$$

dove h è una variabile che rappresenta le distanze e a e b sono i nostri parametri da trovare.

La scelta del solo modello esponenziale è stata fatta per via del fatto che in simulazione si sapeva già il tipo di segnale che veniva generato, ma nulla vieta di usare i modelli presentati in sezione 2.4 e poi cercare la metrica minima tra essi.

Una volta trovati i parametri ottimi a_{Opt} e b_{Opt}, si arriva alla fine dell'algoritmo procedendo alla creazione della nuova matrice di covarianza $\mathbf{\Sigma}^k$ tramite

$$\mathbf{\Sigma}^k_{i,j} = (\sigma_i\sigma_j)a_{Opt}e^{b_{Opt}d(i,j)} \tag{3.6}$$

3.1.1 Pseudocodice

Parametri di input

- $\hat{X}_{k-1}$ è una matrice contenente i valori dei segnali di ogni nodo ricostruiti tramite PCA fino all'istante $k-1$.
- Θ è una matrice contenente le distanze tra un nodo i e un nodo j, ovvero

  $$\Theta[i][j] = \sqrt{(i_x - j_x)^2 + (i_y - j_y)^2}$$

  dove i_x e j_x sono le coordinate x del nodi i e j e i_y e j_y sono le coordinate y sempre del nodo i e j.

Parametri di output

- $\mathbf{\Sigma}$ è la matrice di covarianza stimata tramite il modello

$$\mathbf{\Sigma}_{i,j} = (\sigma_i \sigma_j) a e^{b d_{i,j}}$$

  dove a e b sono due parametri da stimare, $d_{i,j}$ è la distanza tra il nodo i e il nodo j ed infine σ_i e σ_j sono le deviazioni standard del segnale al nodo i e al nodo j.

Variabili locali utilizzate

- N è il numero di nodi della rete.
- min_{Dist} è il minimo tra le distanze di Θ, che si può anticipatamente assumere zero poiché

$$d_{i,j} = 0 \qquad i = j$$

- max_{Dist} è il massimo tra le distanze di Θ

$$max_{Dist} = \max \{\Theta[i][j], \quad \forall\ (i,j)\}$$

- Δ è lo step di quantizzazione delle distanze.
- Θ_Q è la matrice delle distanze quantizzate.
- $\theta_{\mathbf{Q}}$ è il vettore quantizzato di tutte le distanze presenti in Θ_Q ordinate in maniera crescente.
- $\mathbf{\Sigma}^{k-1}$ è la matrice di covarianza dei segnali $\hat{X}_{k-1}$, e $\mathbf{\Sigma}^{k-1}_{i,j}$ è il valore di covarianza tra il nodo i e il nodo j stimata a tempo $k-1$.
- σ è un vettore contenete le deviazioni standard della matrice $\mathbf{\Sigma}^{k-1}$.
- $\mathbf{P}$ è la matrice di correlazione ricavata da $\mathbf{\Sigma}^{k-1}$.
- $\mathbf{f_P}$ è la funzione di correlazione che contiene i valori di correlazione ordinati secondo le distanze quantizzate di $\theta_{\mathbf{Q}}$.
- a_{Opt} e b_{Opt} sono i parametri ottimi stimati dalla funzione di fit.
- **weights** è un vettore di pesi che può essere associato alla funzione di fit.

- **tmp**,**index**,**tmpDistanceVector**,**tmpCorVector**,**par** sono dei vettori temporanei e *index* è uno scalare temporaneo.

Funzioni esterne utilizzate

- index=*indexMin(**vector**)* è una funzione che cerca l'indice di posizione del minimo valore nel vettore passato per argomento.
- **vector**=*reshape(matrix, N,M)* è una funzione che ritorna i valori di una data matrice presi per colonna in una matrice di dimensioni N per M. In questo caso restituisce un vettore di dimensioni $N = num.\ of\ sensors \times num.\ of\ sensors$ e $M = 1$.
- **output**=*find(**vector**==scalar)* è una funzione che ritorna tutti gli indici del vettore **vector** tali per cui la seguente condizione è soddisfatta

$$\mathbf{vector}(i) = scalar$$

- *mean(**vector**)* è una funzione che ritorna la media di tutti i valori del vettore passato come parametro

$$m = \frac{1}{N}\sum_{i=1}^{N} \mathbf{vector}(i) \qquad \text{dove } N = \text{num. di elementi}$$

- *fitMethod(**x**,**y**,**w**)* è una funzione che restituisci i parametri che adattano meglio il modello esponenziale con i dati reali **y** di coordinate **x** con pesi **w**.

Algorithm 3.1.1: SORTANDQUANTIZEDISTANCES(Θ)

comment: Sorting and quantization of the distances

$min_{Dist} \leftarrow \{d^* \in \Theta \mid \forall(i,j),\ d_{i,j} \in \Theta, d_{i,j} \geqslant d^*\}$
$max_{Dist} \leftarrow \{d^* \in \Theta \mid \forall(i,j),\ d_{i,j} \in \Theta, d_{i,j} \leqslant d^*\}$
$\Delta \leftarrow \dfrac{max_{Dist} - min_{Dist}}{N-1}$
$\theta_{\mathbf{Q}} \leftarrow min_{Dist} : \Delta : max_{Dist}$
for $i \leftarrow 1$ **to** N

$$\textbf{do} \begin{cases} \textbf{for } j \leftarrow 1 \textbf{ to } N \\ \quad \textbf{do} \begin{cases} \textbf{tmp} \leftarrow |\theta_{\mathbf{Q}} - \Theta[i][j]| \\ \textbf{index} \leftarrow indexMin(\textbf{tmp}) \\ \Theta_Q[i][j] \leftarrow \theta_{\mathbf{Q}}[\textbf{index}] \end{cases} \end{cases} \qquad \textbf{return } (\Theta_Q, \theta_Q)$$

Algorithm 3.1.2: COVARIANCEESTIMATION($\hat{X}_{k-1}, \Theta$)

$N \leftarrow$ number of sensors

comment: Evaluate quantized distances

res = $sortAndQuantizeDistances(\Theta)$

$\Theta_Q \leftarrow$ **res**(1)

$\theta_{\mathbf{Q}} \leftarrow$ **res**(2)

comment: Generation of the correlation function based on distances

$\mathbf{\Sigma}^{k-1} \leftarrow cov(\hat{X}_{k-1})$

$\sigma \leftarrow \sqrt{diag(\mathbf{\Sigma}^{k-1})}$

for $i \leftarrow 1$ **to** N

$$\textbf{do} \begin{cases} \textbf{for } j \leftarrow 1 \textbf{ to } N \\ \quad \textbf{do } \mathbf{P}_{i,j} \leftarrow \dfrac{\mathbf{\Sigma}_{i,j}^{k-1}}{\sigma_i \sigma_j} \end{cases}$$

tmpDistanceVector $\leftarrow reshape(\Theta_Q, N * N, 1)$

tmpCorVector $\leftarrow reshape(\mathbf{P}, N * N, 1)$

for $i \leftarrow 1$ **to** N

$$\textbf{do} \begin{cases} \textbf{tmp} \leftarrow find(\textbf{tmpDistanceVector} == \theta_{\mathbf{Q}}[i]) \\ \textbf{if tmp} \text{ is not empty} \\ \quad \mathbf{f_P}(i) \leftarrow mean(\textbf{tmpCorVector[tmp]}) \end{cases}$$

comment: Find the best parameters which fit the model

par $\leftarrow fitMethod(\theta_{\mathbf{Q}}, \mathbf{f_P}, \textbf{weghits})$

$a_{Opt} \leftarrow$ **par**[1]

$b_{Opt} \leftarrow$ **par**[2]

comment: Build Sigma based on parameters found

for $i \leftarrow 1$ **to** N

$$\textbf{do} \begin{cases} \textbf{for } j \leftarrow 1 \textbf{ to } N \\ \quad \textbf{do} \begin{cases} \textbf{tmp} \leftarrow |\theta_{\mathbf{Q}} - \Theta[i][j]| \\ index \leftarrow indexMin(\textbf{tmp}) \\ \mathbf{\Sigma}_{i,j} \leftarrow (\sigma_i \sigma_j) a_{Opt} * e^{b_{Opt} \theta_{\mathbf{Q}}[index]} \end{cases} \end{cases}$$

return ($\mathbf{\Sigma}$)

3.2 Windowed Weighted Estimation

A differenza del primo algoritmo, che calcolava il covariogramma considerando la stima di tutti gli istanti temporali precedenti, il secondo algoritmo sfrutta solo l'informazione delle stime più recenti. Il nome WWE è l'acronimo di *Windowed Weighted Estimation* ed è riferito alla versione finestrata e pesata dell'algoritmo *CovarianceEstimation*. La differenza principale tra i due algoritmi risiede appunto nel fatto che si usa solo l'informazione più recente (temporalmente parlando) fornita da $\hat{\mathbf{x}}$, ovvero si considera solo gli ultimi $W_l + W_c - 1$ campioni, dove W_l indica il numero minimo di istanti temporali di $\hat{\mathbf{x}}$ precedenti a k considerati per il calcolo delle matrici di covarianza e W_c il numero di matrici di covarianza distinte che si vogliono calcolare tramite questi campioni.

Difatti, per la stima della matrice di covarianza non si usa una sola matrice di covarianza ma molteplici, calcolate tramite

$$\mathbf{\Sigma}^{k-1,f} = \mathrm{cov}\left(\hat{X}(k - W_l - f + 1 : k - 1)\right) \tag{3.7}$$

dove f identifica la f-esima matrice di covarianza.

Via via che all'aumentare di f si calcolano le matrici di covarianza, queste inglobano dentro di se l'informazione dei valori temporali sempre più vecchi, ovvero per f piccolo la matrice è più imprecisa ma con valori più recenti e per f grande si ha un valore possibilmente più preciso (soprattutto se il segnale è stazionario) ma che ingloba valori più vecchi (cosa negativa nel caso non stazionario).

Un esempio pratico: si assuma di avere $W_l = 2$ e $W_c = 6$. Questo implica che si devono calcolare $W_c = 6$ matrici di covarianza distinte. La prima matrice $\mathbf{\Sigma}^{k-1,1}$ è calcolato utilizzando solo gli ultimi due istanti temporali del segnale relativo a tutti i sensori: $\mathrm{cov}\left(\hat{X}(k-2:k-1)\right)$. La seconda matrice $\mathbf{\Sigma}^{k-1,2}$ è calcolato utilizzando gli ultimi $W_l + f - 1 = 3$ istanti temporali del segnale relativo a tutti i sensori: $\mathrm{cov}\left(\hat{X}(k-3:k-1)\right)$. Così via fino all'ultima matrice $\mathbf{\Sigma}^{k-1,W_c}$ con $W_c = 6$ che viene calcolata usando gli ultimi $W_l + W_c - 1 = 7$ istanti temporali: $\mathrm{cov}\left(\hat{X}(k-7:k-1)\right)$.

Una rappresentazione grafica dell'esempio la si può vedere in Fig. 3.1. L'ascissa indica gli istanti temporali considerati (l'asse è invertito) del segnale $\mathbf{X}$, campionato o stimato, relativo ad un sensore qualsiasi. L'ordinata indica invece l'indice f della

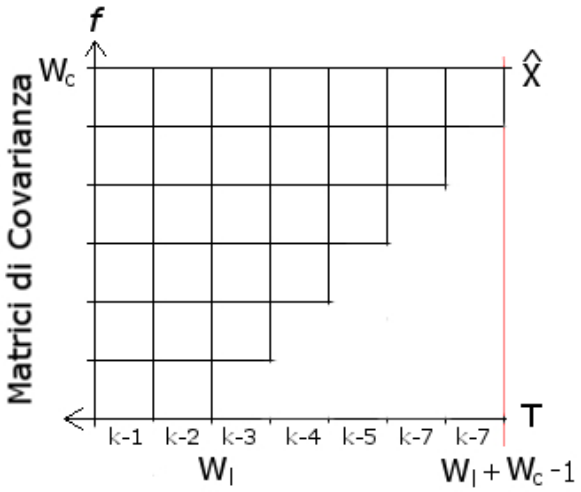


Figura 3.1. *Esempio di finestre $W_l = 2$ e $W_c = 6$*

matrice di covarianza. I quadretti bianchi indicano quali valori di $\mathbf{X}$ sono stati utilizzati per il calcolo della matrice di covarianza. Quindi visto che per $f = 1$ è stato considerato solo il segnale agli ultimi due istanti temporali, vi sono solo due quadretti bianchi. Per $f = 4$ sono invece gli ultimi cinque valori di $\mathbf{X}$ ad essere considerati e quindi sono raffigurati cinque quadretti.

Come prima cosa da notare, è che il valore minimo di W_l è per forza di cose 2 e deriva dal fatto che il numero minino di valori per calcolare la covarianza è 2. Come seconda cosa da notare, è che se si pone $W_c = 1$ e $W_l \geqslant 2$ si ottiene una semplice finestratura del segnale.

La creazione di questo meccanismo complesso di calcolo delle covarianze è sorta dal fatto che si voleva pesare in qualche maniera i valori più recenti del segnale $\hat{\mathbf{x}}$. Se si fossero applicati i pesi direttamente al segnale, se ne avrebbe modificato le ampiezze e la matrice di correlazione risultante sarebbe stata una matrice sbagliata. Per ovviare a questo problema, si è pensato quindi di procedere al calcolo di più matrici di covarianza che contengono l'informazione di diversi istanti temporali, e successivamente pesare quest'ultime nel calcolo della matrice di covarianza finale. A breve si noterà l'applicazione dei pesi.

Tramite queste matrici di covarianza si procede a calcolare esattamente come nel

primo metodo le ora molteplici funzioni di correlazione

$$f_{\mathbf{h}(x)^f} = E[\mathbf{P}^f_{i,j}, \quad \forall\,(i,j) : d(i,j) = h] \tag{3.8}$$

dove

$$\mathbf{P}^f_{i,j} = \frac{\mathbf{\Sigma}^{k-1,f}_{i,j}}{\sigma^f_i \sigma^f_j} \tag{3.9}$$

Anche in questo caso, nella simulazione sono state utilizzate delle distanze quantizzate per il calcolo delle funzioni di correlazione.

Dopo di che, si passa a trovare i parametri a^f_{Opt} e b^f_{Opt} che meglio adattano i W_c modelli alle loro funzioni di correlazione.

Infine si stima la matrice di correlazione $\mathbf{\Sigma}^k$ tramite la formula

$$\mathbf{\Sigma}^k_{i,j} = \sum_{f=1}^{W_c} (\sigma^f_i \sigma^f_j) a^f_{Opt} e^{b^f_{Opt} d(i,j)} w_f \tag{3.10}$$

dove w_f è un determinato peso assegnato alla matrice di correlazione f sotto la condizione che

$$\sum_{f=1}^{W_c} w_f = 1$$

Si vuole ricordare che per soddisfare tale condizione è sufficiente

1. definire in primis un vettore di pesi $\mathbf{w}$ di lunghezza W_c;
2. calcolare la somma dei pesi come

$$\lambda = \sum_{f=1}^{W_c} w_f$$

3. infine dividere il vettore per tale valore

$$\tilde{\mathbf{w}} = \frac{\mathbf{w}}{\lambda}$$

3.2.1 Pseudocodice

Parametri di input

- $\hat{X}$ è una matrice contenente i valori dei segnali di ogni nodo ricostruiti tramite PCA-CS dall'istante $k - (W_l + W_c - 1)$ fino all'istante $k - 1$.

- Θ è una matrice contenente le distanze tra un nodo i e un nodo j, ovvero

$$\Theta[i][j] = \sqrt{(i.x - j.x)^2 + (i.y - j.y)^2}$$

  con $i.x$, $i.y$, $j.x$ e $j.y$ le coordinate x e y del nodo i e j.

- W_l è il numero minimo di letture dei segnali $\hat{X}_{k-1}$ per calcolare le matrici di covarianza.

- W_c è il numero di matrici di covarianza distinte.

Parametri di output

- $\mathbf{\Sigma}$ è la matrice di covarianza stimata tramite il modello

$$\mathbf{\Sigma}_{i,j} = (\sigma_i \sigma_j) a e^{b d_{i,j}}$$

  dove a e b sono due parametri da stimare, $d_{i,j}$ è la distanza tra il nodo i e il nodo j ed infine σ_i e σ_j sono le deviazioni standard del segnale al nodo i e al nodo j.

Variabili locali utilizzate

- N è il numero di nodi della nostra rete.

- $\mathbf{W}$ è una finestra di pesi da applicare alle varie matrici di covarianza.

- Θ_Q è la matrice delle distanze Θ quantizzate.

- $\theta_{\mathbf{Q}}$ è il vettore quantizzato di tutte le distanze presenti in Θ_Q ordinate in maniera crescente.

- $\mathbf{\Sigma}^{(k-1,f)}$ è la matrice di covarianza dei segnali $\hat{X}_{k-(W_l+W_c-1):k-1}$, e $\mathbf{\Sigma}_{i,j}^{(k-1,f)}$ è il valore di covarianza tra il nodo i e il nodo j.

- $\mathbf{\Sigma}^f$ è un vettore contenete le deviazioni standard della matrice $\mathbf{\Sigma}^{(k-1,f)}$.

- $\mathbf{P}^f$ è la matrice di correlazione ricavata da $\mathbf{\Sigma}^{(k-1,f)}$.

- $\mathbf{f_{Pf}}$ è la f-esima funzione di correlazione che contiene i valori di correlazione relativi a $\mathbf{P}^f$ ordinati secondo le distanze quantizzate di $\theta_{\mathbf{Q}}$.

- $\mathbf{a}_{Opt}$ e $\mathbf{b}_{Opt}$ sono vettori conteneti i parametri ottimi stimati dalla funzione di fit per ogni diversa matrice di correlazione.
- **weights** è un vettore di pesi che può essere associato alla funzione di fit.
- **tmp**,**index**,**tmpDistanceVector**,**tmpCorVector**,**par** sono dei vettori temporanei e *index* è uno scalare temporaneo.

Funzioni esterne utilizzate

- $sortAndQuantizeDistances(\Theta)$ è la funzione vista nella sezione precedente che quantizza le distanze di Θ e le ordina in maniere crescente nel vettore θ_Q.
- index=*indexMin(**vector**)* è una funzione che cerca l'indice di posizione del minimo valore nel vettore passato per argomento.
- **vector**=*reshape(matrix, N,M)* è una funzione che ritorna i valori di una data matrice presi per colonna in una matrice di dimensioni N per M. In questo caso restituisce un vettore di dimensioni $N = num.\ of\ sensors \times num.\ of\ sensors$ e $M = 1$.
- **output**=*find(**vector**==scalar)* è una funzione che ritorna tutti gli indici del vettore **vector** tali per cui la seguente condizione è soddisfatta

$$\mathbf{vector}(i) = scalar$$

- *mean(**vector**)* è una funzione che ritorna la media di tutti i valori del vettore passato come parametro:

$$m = \frac{1}{N} \sum_{i=1}^{N} \mathbf{vector}(i)$$

  dove N è il numero di elementi del vettore.
- *fitMethod(**x**,**y**,**w**)* è una funzione che restituisci i parametri che adattano meglio il modello esponenziale con i dati reali **y** di coordinate **x** con pesi **w**.

Algorithm 3.2.1: WWE($\hat{X}, \Theta, W_l, W_c$)

$N \leftarrow$ number of sensors $\quad \mathbf{W} \leftarrow$ correlations weights

comment: Evaluate quantized distances

res = $sortAndQuantizeDistances(\Theta)$

$\Theta_Q \leftarrow \mathbf{res}(1) \quad \theta_{\mathbf{Q}} \leftarrow \mathbf{res}(2)$

comment: Generation of the correlations functions based on distances

$\mathbf{tmpDistanceVector} \leftarrow reshape(\Theta_Q, N * N, 1)$

for $f \leftarrow 1$ **to** W_c

do $\begin{cases} \boldsymbol{\Sigma}^{(k-1,f)} \leftarrow cov(\hat{X}[k - W_l - f + 1 : k - 1]) \\ \boldsymbol{\Sigma}^f \leftarrow \sqrt{diag(\boldsymbol{\Sigma}^{(k-1,f)})} \\ \textbf{for } i \leftarrow 1 \textbf{ to } N \\ \quad \textbf{do} \begin{cases} \textbf{for } j \leftarrow 1 \textbf{ to } N \\ \quad \textbf{do } P^f_{i,j} \leftarrow \dfrac{\boldsymbol{\Sigma}^{(k-1,f)}_{i,j}}{\boldsymbol{\Sigma}^f_i \boldsymbol{\Sigma}^f_j} \end{cases} \end{cases}$

for $i \leftarrow 1$ **to** N

do $\begin{cases} \mathbf{tmp} \leftarrow find(\mathbf{tmpDistanceVector} == \theta_{\mathbf{Q}}[i]) \\ \textbf{for } f \leftarrow 1 \textbf{ to } W_c \\ \quad \textbf{do} \begin{cases} \mathbf{tmpCorVector} \leftarrow reshape(P^f, N * N, 1) \\ \textbf{if } \mathbf{tmp} \text{ is not empty} \\ \quad \mathbf{f_{Pf}}(i) \leftarrow mean(\mathbf{tmpCorVector[tmp]}) \end{cases} \end{cases}$

comment: Find the best parameters which fit the model

for $f \leftarrow 1$ **to** W_c

do $\begin{cases} \mathbf{par} \leftarrow fitMethod(\theta_{\mathbf{Q}}, \mathbf{f_{Pf}}, \mathbf{weghits}) \\ \mathbf{a}_{Opt}(f) \leftarrow \mathbf{par}[1] \quad \mathbf{b}_{Opt}(f) \leftarrow \mathbf{par}[2] \end{cases}$

comment: Build Sigma based on parameters found

for $i \leftarrow 1$ **to** N

do $\begin{cases} \textbf{for } j \leftarrow 1 \textbf{ to } N \\ \quad \textbf{do} \begin{cases} \mathbf{tmp} \leftarrow |\theta_{\mathbf{Q}} - \Theta[i][j]| \\ index \leftarrow indexMin(\mathbf{tmp}) \\ \textbf{for } f \leftarrow 1 \textbf{ to } W_c \\ \quad \textbf{do } \boldsymbol{\Sigma}_{i,j} \leftarrow \boldsymbol{\Sigma}_{i,j} + \mathbf{W}(f)(\boldsymbol{\Sigma}^f_i \boldsymbol{\Sigma}^f_j)\mathbf{a}_{Opt}(f) e^{\mathbf{b}_{Opt}(f)\theta_{\mathbf{Q}}[index]} \end{cases} \end{cases}$

return $(\boldsymbol{\Sigma})$

3.3 Funzioni di fit

Per la stima dei parametri del modello sono stati testati tre metodi di fit diversi. Il primo basato sulle funzioni base di Matlab. Gli ultimi due sono basati sul metodo di Brent per la ricerca di minimo e sono stati sviluppati in C++. Il metodo di Matlab, sebbene più facile ed immediato da usare, è molto più lento delle sue controparti in C++. I tempi computazionali del primo infatti possono rallentare il singolo processo di anche 10 volte il tempo totale di esecuzione, mentre gli ultimi due hanno tempi del tutto trascurabili rispetto al tempo di esecuzione dell'intera simulazione. Si è quindi voluto prestare maggiore attenzione agli ultimi due.

3.3.1 Brent1

La funzione che l'algoritmo Brent1 ottimizza è la seguente

$$f(b) = \frac{1}{N} \sum_i \left(a e^{b\mathbf{x}(i)} - \mathbf{y}(i) \right)^2 \tag{3.11}$$

dove $\mathbf{x}$ e $\mathbf{y}$ sono i vettori contenenti rispettivamente le distanze e il valor medio di correlazione a tal distanza. Trattandosi di correlazione, si è assunta inizialmente la variabile $a = 1$ poiché la correlazione a distanza nulla implica la correlazione che un nodo ha con se stesso ed è quindi unitaria. A questo punto l'unica variabile da stimare è b.

Parametri di input

- $\mathbf{x}$ è il vettore delle distanze.
- $\mathbf{y}$ è il vettore dei valori medi di correlazione alle distanze $\mathbf{x}$.

Parametri di output

- $[a_{Opt},\ b_{Opt}]$ è un vettore contenente le due variabili a e b che ottimizzano la funzione di correlazione.

Variabili locali utilizzate

- A è il limite inferiore dell'intervallo in cui cercare la nostra variabile che ottimizza la funzione.

- B è il limite superiore dell'intervallo in cui cercare la nostra variabile che ottimizza la funzione.
- δ è la precisione con cui vogliamo cercare la nostra variabile.
- **res** è un vettore temporaneo.

Funzioni esterne utilizzate

- *brentOptimizationCpp* è l'algoritmo principale standard Brent che minimizza la funzione (3.11) implementato in C++.

Algorithm 3.3.1: BRENT1($\mathbf{x}, \mathbf{y}$)

comment: Setting of parameters

$A \leftarrow$ lower bound

$B \leftarrow$ upper bound

$\delta \leftarrow$ required precision

comment: Brent optimization

res $\leftarrow$ *brentOptimizationCpp*$(A, B, \delta, \mathbf{x}, \mathbf{y})$

$b_{Opt} \leftarrow$ **res**(1)

$a_{Opt} \leftarrow 1$;

return (a_{Opt}, b_{Opt})

3.3.2 Brent2

Si è notato che all'inizio della simulazione l'algoritmo di fit di Matlab stimava valori inferiori a 1 per la variabile a del modello, e poi convergeva a 1. Il motivo di ciò è che all'inizio si hanno pochi dati per fare la stima corretta del modello e certe volte si ha un errore di stima con valori di a inferiori a 1. Questo implica che nella fase iniziale dell'algoritmo c'è l'eventualità che un modello con il parametro $a = 1$ si adatti ai dati reali con un errore più grande di un modello con il parametro a diverso da uno e di conseguenza introducendo un errore maggiore.

Si è fatto quindi una modifica all'algoritmo precedentemente presentato, dove ora la variabile a non è più fissa ma viene anch'essa ottimizzata con Brent. Data la funzione

da minimizzare

$$f(a,b) = \frac{1}{N}\sum_i \left(ae^{b\mathbf{x}(i)} - \mathbf{y}(i)\right)^2 \tag{3.12}$$

l'algoritmo stima inizialmente un valore b_{Opt} con a fisso a 1 e considerando b come variabile. Una volta trovato il valore b_{Opt} stima il valore di a_{Opt} tenendo fisso $b = b_{Opt}$ e considerando a come variabile. Il processo viene cosi ripetuto finché la distanza tra i due vettori formati uno dalle variabili correnti e uno dalle variabili al passo successivo non raggiunge una certa soglia

$$\sqrt{(a_{(Opt,k)} - a_{(Opt,k-1)})^2 + (b_{(Opt,k)} - b_{(Opt,k-1)})^2} \leqslant \Gamma \tag{3.13}$$

Parametri di input

- $\mathbf{x}$ è il vettore delle distanze.
- $\mathbf{y}$ è il vettore dei valori medi di correlazione alle distanze $\mathbf{x}$.

Parametri di output

- $[a_{Opt},\ b_{Opt}]$ è un vettore contenente le due variabili a e b che ottimizzano la funzione di correlazione.

Variabili locali utilizzate

- A è il limite inferiore dell'intervallo in cui cercare la nostra variabile che ottimizza la funzione.
- B è il limite superiore dell'intervallo in cui cercare la nostra variabile che ottimizza la funzione.
- δ è la precisione con cui vogliamo cercare la nostra variabile.
- Γ è la soglia minima di spostamento delle variabili a_{Opt} e b_{Opt}.
- *flag* è una variabile utilizzata per uscire dal ciclo while nel caso in cui la soglia (3.13) sia stata raggiunta.
- *ctrl* e *ctrlMax* sono rispettivamente una variabile ed una costante di controllo aggiunto per evitare che il ciclo while iteri un numero eccessivo di volte rallentando la simulazione.

- **res** è un vettore temporaneo.

- a_{Old} e b_{Old} sono le variabili utilizzare per tenere in memoria i valori dei parametri a_{Opt} e b_{Opt} al passo precedente.

Funzioni esterne utilizzate

- $brentOptimizationCpp(A, B, \delta, \mathbf{x}, \mathbf{y}, C, bool)$ è l'algoritmo di Brent che minimizza la funzione (3.12) implementato in C++. Il parametro *bool* è usato per indicare se bisogna ottimizzare rispetto alla variabile a ($bool = 1$) o rispetto alla variabile b ($bool = 0$) mentre il parametro C è utilizzato per assegnare il valore costante ad a o a b a seconda di quelle variabile si è intenzionati ad ottimizzare.

Algorithm 3.3.2: BRENT2($\mathbf{x}, \mathbf{y}$)

comment: Setting of parameters

$A \leftarrow$ lower bound

$B \leftarrow$ upper bound

$\delta \leftarrow$ required precision

$\Gamma \leftarrow$ fixed threshold

$flag \leftarrow true$

$ctrl \leftarrow 0$

$ctrlMax \leftarrow 30$

comment: Initialization

$\mathbf{res} \leftarrow brentOptimizationCpp(A, B, \delta, \mathbf{x}, \mathbf{y}, 1, 0)$

$b_{Opt} \leftarrow \mathbf{res}(1)$

$b_{Old} \leftarrow b_{Opt}$

$\mathbf{res} \leftarrow brentOptimizationCpp(A, B, \delta, \mathbf{x}, \mathbf{y}, b_{Old}, 1)$

$a_{Opt} \leftarrow \mathbf{res}(1)$

$a_{Old} \leftarrow a_{Opt}$

comment: While process

while $flag$ && $ctrl \leqslant ctrlMax$

do
- $ctrl++; \mathbf{res} \leftarrow brentOptimizationCpp(A, B, \delta, \mathbf{x}, \mathbf{y}, a_{Old}, 0)$
- $b_{Opt} \leftarrow \mathbf{res}(1)$
- $b_{Old} \leftarrow b_{Opt}$
- $\mathbf{res} \leftarrow brentOptimizationCpp(A, B, \delta, \mathbf{x}, \mathbf{y}, b_{Old}, 1)$
- $a_{Opt} \leftarrow \mathbf{res}(1)$
- $a_{Old} \leftarrow a_{Opt}$
- **if** $\sqrt{(a_{Opt} - a_{Old})^2 + (b_{Opt} - b_{Old})^2}$
  - **then** $flag = false$
  - **else**
    - $a_{Old} \leftarrow a_{Opt}$
    - $b_{Old} \leftarrow b_{Opt}$

return (a_{Opt}, b_{Opt})

CAPITOLO 4

RISULTATI

Nel capitolo verranno presentati i risultati ottenuti dallo studio di:

- differenze sull'utilizzo di matrice di correlazione o covarianza da passare al PCA;
- variabili in gioco nell'algoritmo WWE (W_l e W_c);
- comparazione dei due algoritmi con il DNS;
- comparazione con altri algoritmi;

Gli altri algoritmi utilizzati sono

- il *Lightweight Temporal Compression* (LTC) [8];
- un metodo basato sulla *Discrete Cosine Transform* [8];
- il *Deterministic Node Selection* [9];
- il *Distributed Source Coding* (DSC) [10] [11].

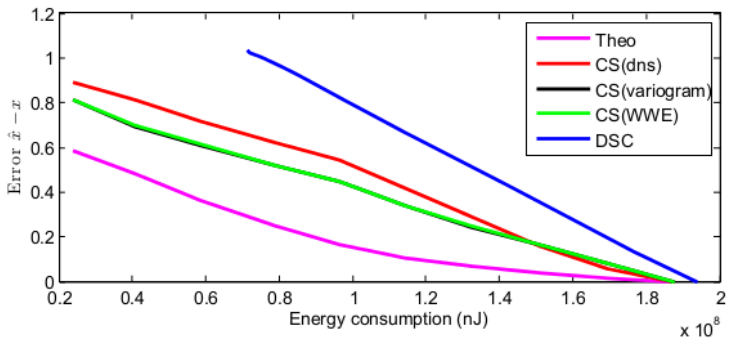
Le coppie di valori di correlazione base utilizzate sono state (prendendo spunto dai *papers* precedentemente citati) le seguenti poiché indicano dei casi abbastanza diversi tra loro.

	ρ_t	ρ_s
1.	0	0.001
2.	0.8	0.001
3.	0.98	0.001
4.	0.5	5

Con la notazione ρ_t si vuole indicare la correlazione temporale del segnale e con ρ_s la sua correlazione spaziale. A differenza di ρ_t che è compreso tra $[-1, 1]$, il parametro ρ_s può assumere un qualsiasi valore reale positivo, poiché esso indica la distanza spaziale a cui il modello di correlazione del segnale utilizzato pre la creazione del segnale sintetico raggiunge il valore 0.05.

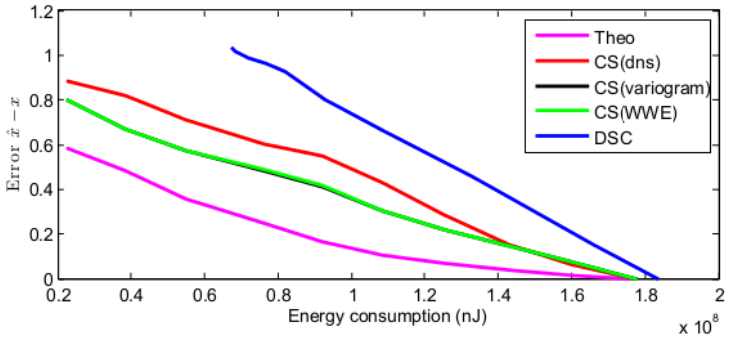
4.1 Differenze tra matrice di covarianza e matrice di correlazione

Dopo svariati tentativi si è notata una leggera differenza tra l'utilizzo della matrice di covarianza e correlazione da dare in pasto al PCA. Sebbene la differenza sia minima, sembra che ci sia un risultato leggermente migliore utilizzando come previsto la matrice di covarianza (si prenda come esempio in Fig. 4.1 e Fig. 4.2), eccezion fatta nel caso in cui la correlazione spaziale sia grande, come in Fig. 4.3 e Fig. 4.4.



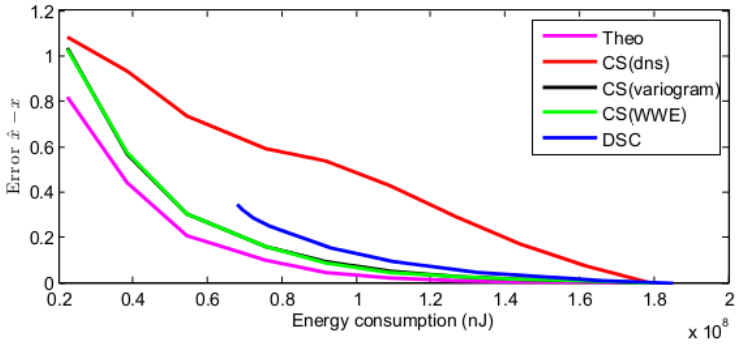
(a) *Correlazione*

Figura 4.1. $\rho_t = 0.98$ $\rho_s = 0.001$



(a) *Covarianza*

Figura 4.2. $\rho_t = 0.98$ $\rho_s = 0.001$



(a) *Correlazione*

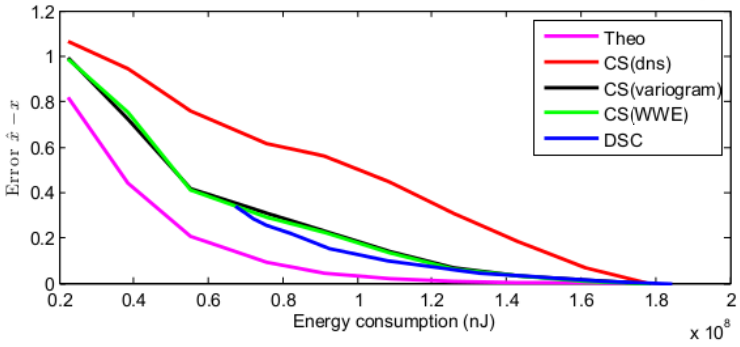
Figura 4.3. $\rho_t = 0.5$ $\rho_s = 5$

4.2 Studio di W_l e W_c

Per quanto concerne lo studio delle grandezze delle due finestre W_l e W_c, si è studiato l'errore in funzione dell'aumentare di una delle due finestre, tenendo fisso il valore dell'altra. In particolare si è scoperto che

- la grandezza della finestra W_c è irrilevante nel nostro caso. Difatti il grafico risultante era pressoché piatto.

- la grande della finestra W_l è stata presa di 40 unità temporali poiché la curva relativa alla versione finestrata dell'algoritmo aveva un andamento decrescente e



(a) *Covarianza*

Figura 4.4. $\rho_t = 0.5$ $\rho_s = 5$

raggiungeva con buona approssimazione il limite definito dalla sua controparte non finestrata attorno a quel valore.

4.3 Comparazione degli algoritmi VarEst, WWE e DNS

Per la loro comparazione sono state prese in esame due metriche:

- la differenza tra le matrici di covarianza Σ in funzione del tempo;
- l'errore di $\hat{x}$ in funzione del tempo.

Ovvero la prima è una funzione del tipo

$$f_\Sigma(t) = \left|\Sigma_{th} - \Sigma(t)\right| \tag{4.1}$$

dove Σ_{th} è la matrice di correlazione teorica calcolata dai segnali originali e $\Sigma(t)$ è la matrice calcolata dall'algoritmo campionando e stimando nel tempo fino al momento t. La seconda è l'errore di stima del segnale $\hat{x}$ rilasciato dal PCA in confronto con il segnale originale al medesimo istante:

$$f_{\hat{x}}(t) = \frac{\sum_{i=1}^{N} \left|x_i(t) - \hat{x}_i(t)\right|}{N} \tag{4.2}$$

dove N è il numero di sensori, $x_i(t)$ è il segnale originale al sensore i al tempo t e $\hat{x}_i$ è il suo analogo stimato.

Lo studio in caso stazionario era abbastanza irrilevante poiché si potevano notare gli stessi risultati che si potevano vedere nella prima parte del caso non stazionario, quindi si è voluto proporre solo quest'ultimo: in pratica guardando nel primo quarto di ascissa delle figure riportate, si può notare il comportamento in caso stazionario. Dalle figure riportate si può notare come i metodi basati sul covariogramma preformino meglio del DNS, e come la versione finestrata raggiunga spesso il *lower bound* teorico. Nell'errore del segnale in Fig. (4.6) è importante notare che la versione finestrata converge velocemente al *lower bound* (50 unità temporali) rispetto alla versione non finestrata.

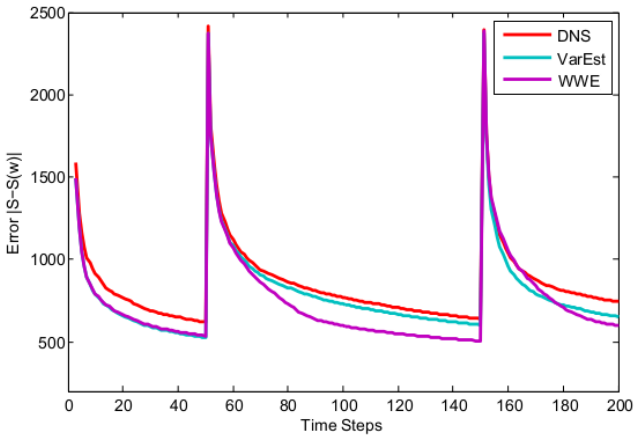


Figura 4.5. $|\Sigma - \hat{\Sigma}|$

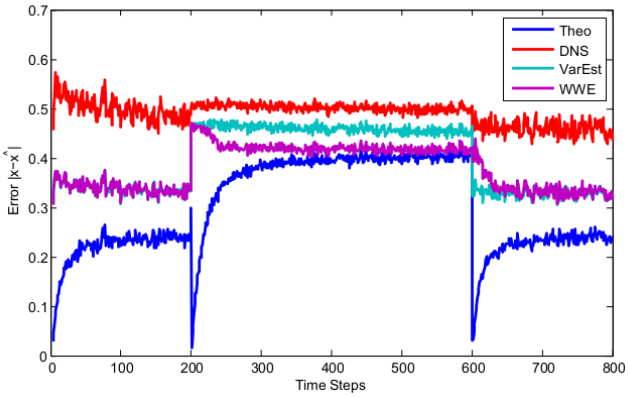


Figura 4.6. $|x - \hat{x}|$

Per la creazione del segnale sono stati utilizzati due segnali distinti con diverse correlazioni e poi sono stati salvati in un unico grande segnale come in Fig. 4.7: metà del primo segnale all'inizio, il secondo segnale al centro ed infine l'ultima metà del

primo segnale alla fine. I parametri utilizzati sono stati

Caso	$\rho_{t,1}$	$\rho_{s,1}$	$\rho_{t,2}$	$\rho_{s,2}$
1	0	5	0.98	0.001
2	0.9	5	0.1	0.001
3	0.98	5	0.5	5

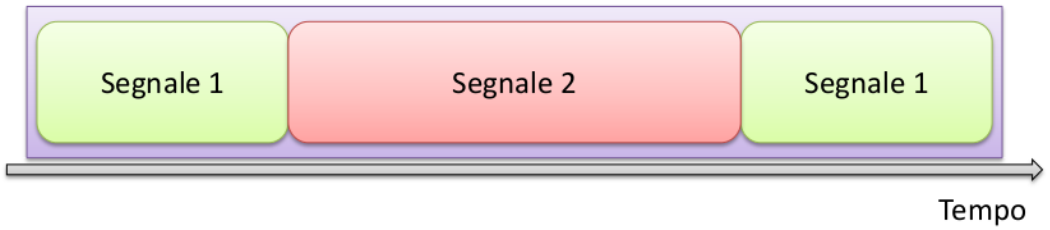


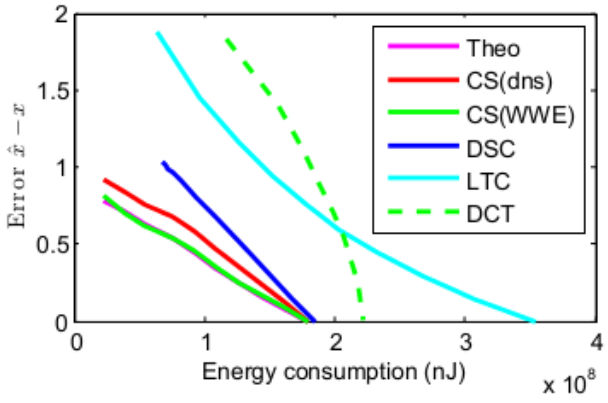
Figura 4.7. *Esempio segnale non stazionario*

4.4 Comparazione di tutti gli algoritmi

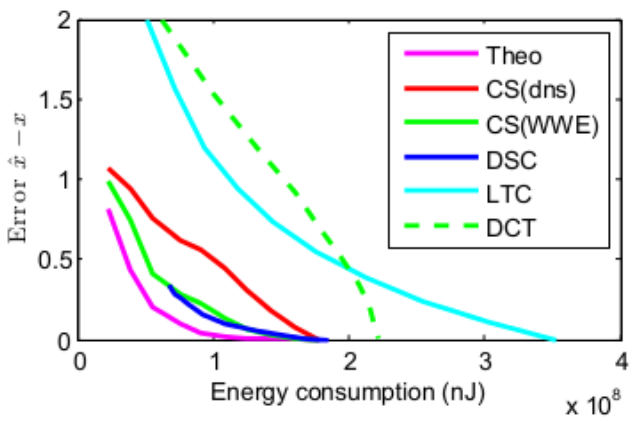
Infine nei seguenti grafici si può vedere la versione finestrata dell'algoritmo presentato in questa tesi in confronto con altri algoritmi precedentemente studiati in [8] [9] [10] [11]. Si è presentato solo la versione finestrata perché la prima versione non è implementabile in termini pratici: è impossibile memorizzare tutto lo storico dei segnali all'interno dei sensori e sarebbe valido solo in caso di segnale stazionario, cosa assai rara. I grafici presentano l'errore di stima in funzione del consumo energetico della rete che è il principale obiettivo di questo lavoro.

Si nota subito come l'algoritmo WWE performi meglio degli altri algoritmi presi in esame, ad eccezion fatta del caso di grande correlazione temporale ove l'algoritmo LTC ha migliori risultati: questo è dovuto al fatto che il suo metodo è basato unicamente sulla correlazione temporale mentre il metodo del covariogramma è basato sia sulla correlazione spaziale che su quella temporale.

Un ulteriore esempio di questa differenza si evince dallo studio dei segnali dello stadio Sant Jordi di Barcellona (Fig. 4.10), che hanno una forte correlazione temporale. Per lo studio dei segnali è stato preso in esame un set di campioni di più di 6000 istanti temporali dove la distanza fisica tra ogni istante temporale corrisponde a 5 secondi. Da



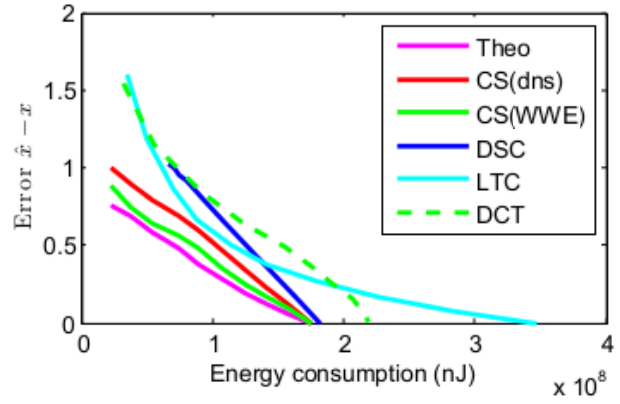
(a) $\rho_t = 0$ $\rho_s = 0.001$



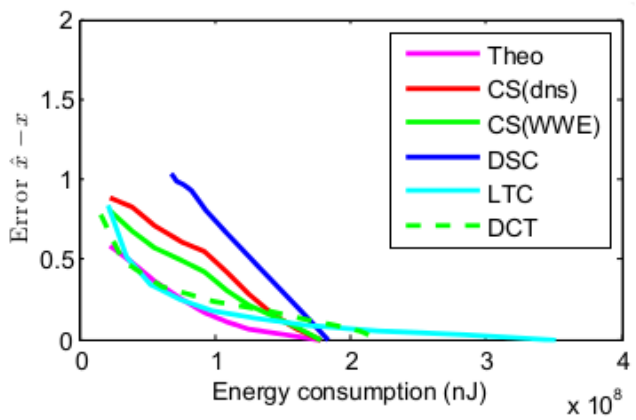
(b) $\rho_t = 0.5$ $\rho_s = 5$

Figura 4.8. *Esempi caso stazionario*

come è stata implementata la simulazione, si è assunto che il *sensing* completo da parte di tutti i sensori sia rifatto ogni $T = [\ 300,\ 600,\ 1000]$ istanti temporali. In pratica ogni T istanti tutti i sensori prelevano il segnale per i due istanti successivi in modo da fare prevenire in parte l'errore accumulativo che si porta dietro l'algoritmo WWE.



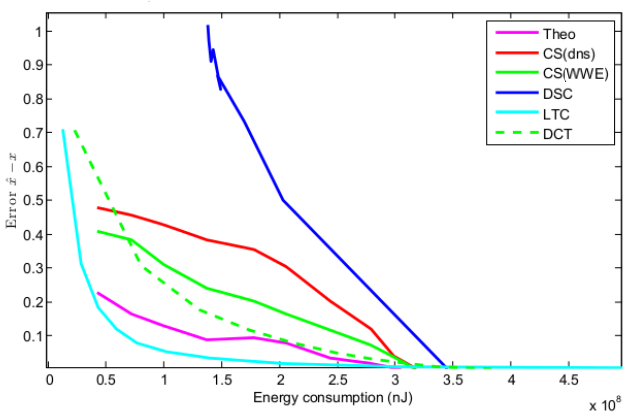
(a) $\rho_t = 0.8$ $\rho_s = 0.001$



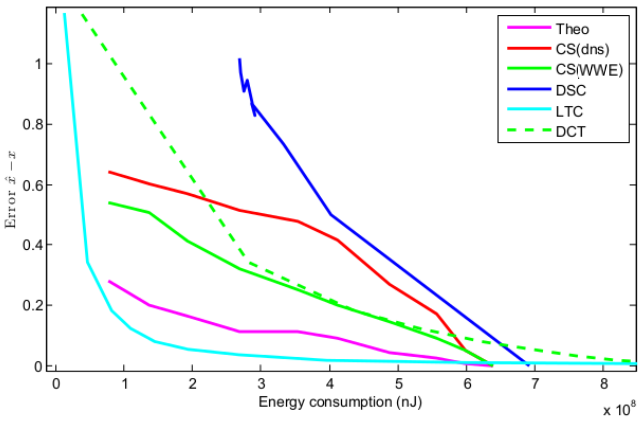
(b) $\rho_t = 0.98$ $\rho_s = 0.001$

Figura 4.9. *Esempi caso stazionario*

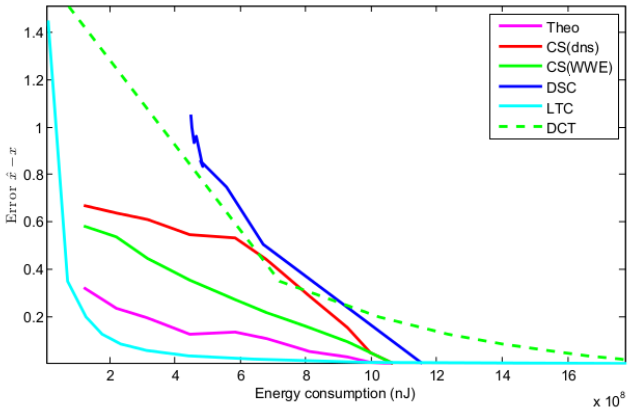
Infine si è voluto studiare anche il caso non stazionario. Dai grafici si può notare che come nel caso stazionario, il metodo WWE performi meglio degli altri metodi, tranne nel caso di correlazione temporale forte in ambo i segnali (Caso 3) che i risultati del LTC e DCT sono molto simili al WWE. In questo caso è da notare che le prestazioni dell'algoritmo sono leggermente peggiorate, infatti esso ha raggiunto il DSC e in nessuno dei casi raggiunge il *lower bound* teorico.



(a) $T = 300$

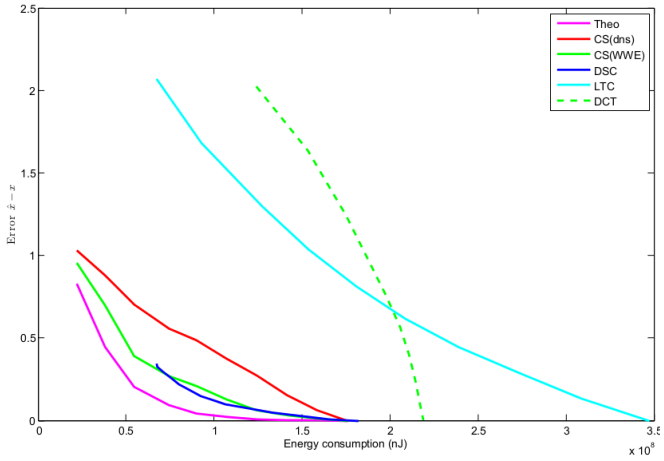


(b) $T = 600$

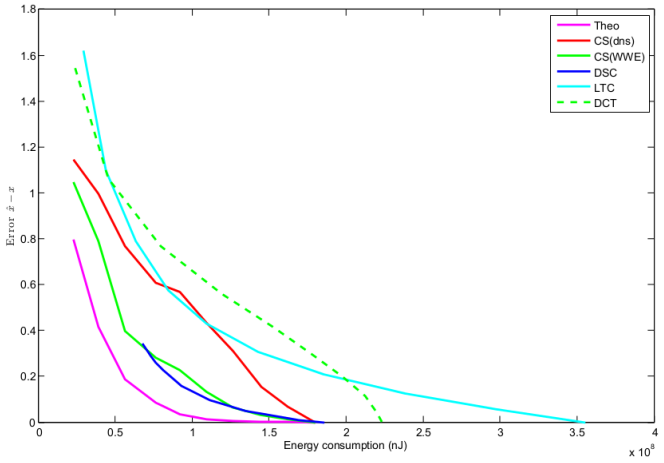


(c) $T = 1000$

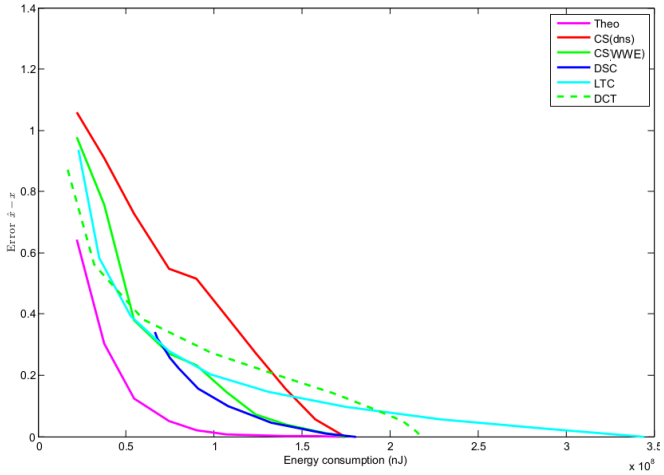
Figura 4.10. *Dati Sant Jordi*



(a) *Caso 1*



(b) *Caso 2*



(c) *Caso 3*

Figura 4.11. *Caso non stazionario*

CAPITOLO 5

CONCLUSIONI E SVILUPPI FUTURI

5.1 Conclusioni

Dopo una breve introduzione agli strumenti utilizzati (l'algoritmo di compressione spazio-temporale PCA-CS, il concetto di covariogramma e i tipi di segnali utilizzati), dopo lo studio della stima della funzione di correlazione e dopo l'*overview* dei risultati, si può concludere che la stima della funzione di covariogramma produce notevoli risultati in quasi tutte le condizioni di correlazione del segnale. Infatti in caso di bassa e media correlazione temporale ed indipendentemente dalla correlazione spaziale, la stima del covariogramma ha risultati migliori di tutti gli altri algoritmi con cui è stata comparata. In più, nel caso di correlazioni temporali e spaziali pressoché nulle, questo algoritmo raggiunge l'ottimo possibile toccando il *lower bound*. Gli unici in cui si ottengono risultati non ottimi, ma comunque buoni, sono quelli con alta correlazione temporale, dove l'algoritmo WWE è secondo solo all'algoritmo LTC, che ricordiamo essere un algoritmo basato unicamente sulla correlazione temporale.

5.2 Sviluppi futuri

Possibili sviluppi futuri per raggiungere il *lower bound* in tutti i possibili casi di correlazioni, è quello di migliorare la selezione dei sensori per il campionamento, poiché per ora esso viene fatto in maniera casuale o in modo *round robin*, mentre sarebbe

meglio l'utilizzo di una selezione dei sensori basata sulla correlazione tra sensori e/o sul consumo energetico del singolo sensore.

In oltre, altri possibili sviluppi futuri sono quelli di adattare il *framework* a casi reali in campi come la meteorologia, la biomedica, lo studio dell'inquinamento ambientale e acustico.

Un esempio è l'utilizzo dei dispositivi mobili quali *smartphone* e *tablet* che sempre più spesso si vedono in circolazione. Infatti se ogni dispositivo mobile in circolazione fosse dotato di sensori in grado di percepire i dati di interesse si potrebbe disporre di una quantità rilevante di dati da analizzare in maniera gratuita. Ovviamente, essendo i dispositivi mobili alimentati a batteria, la principale preoccupazione risiede nel fatto che questi sensori consumino minor energia possibile in modo da non procurare disagi all'utente del dispositivo.

BIBLIOGRAFIA

[1] Ian F. Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, e Erdal Cayirci, *A Survey on Sensor Networks*, IEEE Communications Magazine, pp. 102-114 , Agosto 2002

[2] Jonathon Shlens , *A Tutorial on Principal Component Analysis*, 22 Aprile 2009

[3] G. Quer, R. Masiero, G. Pillonetto, M. Rossi e M. Zorzi, *Sensing, Compression, and Recovery for WSNs: Sparse Signal Modeling and Monitoring Framework*, IEEE Transactions on Wireless Communications, Vol. 11, No. 10, Ottobre 2012

[4] Muhammad Umer, Lars Kulik, Egemen Tanin, *Spatial interpolation in wireless sensor networks: localized algorithms for variogram modeling and Kriging*, Geoinformatica, Vol. 14, pp. 101–134, 2010

[5] T. Gneiting, Z. Sasvári, M. Schlather, *Analogies and Correspondences Between Variograms and Covariance Functions*, NRCSE Technical Report Series , No. 056, 12 Ottobre 2000

[6] Davide Zordan, Giorgio Quer, Michele Zorzi and Michele Rossi, *Modeling and Generation of Space-Time Correlated Signals for Sensor Network Fields*, IEEE GLOBECOM 2011, Houston, Texas, US, 5-9 Dicembre 2011.

[7] Sara A. Van De Geer, *Least Squares Estimation*, Volume 2, pp. 1041–1045, 2005

[8] Davide Zordan, Borja Martinez, Ignasi Villajosana and Michele Rossi, *On the Performance of Lossy Compression Schemes for Energy Constrained Sensor Networking*, ACM Transactions on Sensor Networks, Vol. 11, No. 1, Agosto 2014.

[9] Michele Rossi, Mohsen Hooshmand, Davide Zordan, Michele Zorzi, *Evaluating the Gap Between Compressive Sensing and Distributed Source Coding in WSN*, IEEE International Conference on Computing, Networking and Communications (ICNC), February 16-19, Anaheim, California, US, 2015.

[10] S. S. Pradhan e K. Ramchandran, *Distributed Source Coding Using Syndromes (DISCUS): Design and Construction*, IEEE Transactions on Information Theory, vol. 49, no. 3, Mar. 2003

[11] Z. Xiong, A. D. Liveris, and S. Cheng, *Distributed Source Coding for Sensor Networks*, IEEE Signal Processing Magazine, vol. 21, no. 5, Sep. 2004.

[12] E. Candés, J. Romberg, e T. Tao, *Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information* IEEE Trans. Inf. Theory, vol. 52, no. 2, pp. 489–509, Feb. 2006.

[13] Denis Allard, *Geostatistical Classification and Class Kriging*, Journal of Geographic Information and Decision Analysis, vol. 2, no. 2, pp. 77-90, 1998

[14] Giorgio Quer, Riccardo Masiero, Daniele Munaretto, Michele Rossi, Joerg Widmer and Michele Zorzi, *On the Interplay Between Routing and Signal Representation for Compressive Sensing in Wireless Sensor Networks*, Workshop on Information Theory and Applications, ITA 2009, San Diego, CA, US, Feb. 8-13, 2009.

RINGRAZIAMENTI

Desidero ricordare tutti coloro che mi hanno aiutato nella stesura della tesi con suggerimenti, critiche ed osservazioni: a loro va la mia gratitudine, anche se a me spetta la responsabilità per ogni errore contenuto in questa tesi.

Ringrazio anzitutto il professor Michele Rossi, Relatore, ed il ricercatore Mohsen Hooshmand, Co-relatore che senza il loro supporto ed aiuto questa tesi non esisterebbe.

Proseguo con tutti i ragazzi del Signet per i loro suggerimenti e critiche che mi hanno permesso di migliorare la tesi sotto aspetti sia pratici che estetici.

Un ringraziamento particolare va ai colleghi ed agli amici che mi hanno supportato o che hanno speso parte del proprio tempo per leggere e discutere con me le bozze del lavoro.

Vorrei infine ringraziare le persone a me più care: i miei amici, la mia famiglia ed infine la mia fidanzata, a cui questo lavoro è dedicato.