Università degli studi di Padova

Dipartimento di Ingegneria dell'informazione
Laurea Magistrale in Ingegneria Informatica

Ricerca di anomalie su serie temporali di temperature basata su rappresentazione simbolica

Studente:
Francesco Carbone
1034752

Relatore:
Prof. C. Pizzi

Correlatore:
Prof. A. Cenedese

Anno Accademico 2013/2014

Un sentito ringraziamento ai miei genitori e a mia sorella che, con il loro incrollabile sostegno morale ed economico, mi hanno permesso di raggiungere questo importante obiettivo. Alla mia ragazza e compagna di vita, per essermi stata accanto durante le mie lunghe sessioni di studio con l'amore che solo chi crede in un futuro concreto assieme può riuscire a dimostrare. La ringrazio soprattutto per l'orgoglio che ha riposto in tutti i traguardi finora raggiunti assieme. Desidero inoltre ringraziare i miei relatori, la professoressa Pizzi ed il professor Cenedese, che hanno creduto nel mio lavoro al punto da proporne la pubblicazione. Ai miei amici, per essermi stati vicini nei lunghissimi periodi di studio. Ringrazio con affetto Frezz, Giulio e Gabriele, per aver condiviso con me gli ultimi estenuanti esami, sono stati per me grandi amici oltre che semplici compagni.

Sommario

L'analisi dei dati per il monitoraggio di attività umane svolge un ruolo cruciale nello sviluppo di politiche intelligenti atte a migliorare il benessere e la sostenibilità delle nostre città. Tra le diverse applicazioni in questo contesto si colloca lo studio di anomalie a livelli giornalieri e settimanali di dati forniti da sensori di diversa natura.
Per questo lavoro è stato sviluppato un approccio di *anomaly detection* basato su tecniche di *clustering* e di consenso che sfrutta la potenza della discretizzazione SAX e la specificità delle serie temporali da analizzare. Più in dettaglio, verrà discusso un *tool* sviluppato capace di costruire un modello per il comportamento previsto, composto da uno o più *pattern* significativi, del segnale analizzato; tutte le sottosequenze sufficientemente distanti da esso verranno infine etichettate come anomale. Tale modello può essere esportato ed utilizzato per analisi successive. Una classificazione gerarchica facoltativa delle anomalie individuate potrà, inoltre, aiutare a scoprire le cause che hanno portato a tali comportamenti inaspettati.
È interessante come la costruzione del modello sia realizzata mediante una procedura di "proposta e voto" tra il risultato dell'esecuzione di diversi *k-means clustering* indipendenti. Inoltre, per supportare la fase di analisi, è possibile utilizzare un calendario come fonte supplementare di informazione allo scopo di discriminare al meglio tra le anomalie davvero indesiderate e quelle aspettate (per esempio nei week-end). L'approccio, infine, supporta due modalità di funzionamento: supervisionato e non supervisionato.
Negli esperimenti realizzati (analisi delle temperature con l'obiettivo di evitare sprechi di energia), lo strumento implementante l'approccio proposto è stato in grado di rilevare tutte le anomalie attese e, talvolta, alcune anomalie inaspettate (ma pur sempre tali). Il *software* è stato testato con e senza un calendario di riferimento e su diversi *dataset* liberamente disponibili, con risultati simili.

Indice

Capitolo 1

Introduzione

Grazie ai recenti progressi della tecnologia ed a nuovi paradigmi nella teoria dei sistemi distribuiti, al giorno d'oggi le reti di sensori ed attuatori hanno cambiato radicalmente il nostro modo di interagire con l'ambiente circostante. Questi dispositivi riescono ad offrire accesso ad una grandissima quantità di informazioni di qualità senza precedenti, che possono rivoluzionare la nostra capacità di controllare lo spazio umano. Le applicazioni che derivano da questa rivoluzione tecnologica, che ha beneficiato della sinergia tra diverse discipline scientifiche, coprono una vasta gamma di settori ed obiettivi: miglioramento della produttività in un contesto industriale, assistenza nelle azioni di vita quotidiana, monitoraggio intelligente degli spazi pubblici con particolare attenzione alla *privacy* dell'utenza. In particolare, i progressi nella analisi dei dati per monitorare le attività umane connesse svolgono un ruolo cruciale nello sviluppo di politiche intelligenti per migliorare il benessere e la sostenibilità delle nostre città.

Basandosi su tali premesse, questo lavoro è motivato dal problema di rilevamento delle anomalie in serie temporali di temperature ottenute attraverso una rete di sensori distribuiti all'interno del perimetro della scuola elementare Rigoni. Lo scopo di tale applicazione è quello di individuare eventuali sprechi energetici, un problema comune nel contesto di studio su edifici intelligenti. È interessante come questo tipo di serie temporali ha la specificità di essere strettamente in correlazione con le attività umane: infatti uno dei fattori principali che influenza i valori di temperatura misurata all'interno di una stanza, oltre al cambiamento climatico stagionale, è la presenza di persone. Una fonte aggiuntiva di informazioni, sotto forma di calendario scolastico, ha permesso di rilevare anomalie anche in quelle situazioni in cui l'andamento delle temperature apparisse *standard*. Per esempio, se in un periodo festivo la serie possiede un comportamento riconosciuto come "normale" è possibile che in realtà esso non sia da riconoscere come tale, in quanto ci si aspetta che in questi particolari periodi i dati rilevati mutino in funzione dell'assenza di personale nelle aule monitorate. L'approccio proposto in questo lavoro si basa su una rappresentazione simbolica delle serie storiche e su di una procedura iterativa in cui un campionamento casuale dei dati da analizzare consente di costruire un modello necessario al rilevamento delle anomalie. Mediante una procedura di voto, ispirata a classici algoritmi basati su consenso atti ad analizzare grandi quantità di dati rumorosi [1], viene infine determinata la migliore coppia modello-anomalie individuata a ciascuna iterazione. Come verrà dimostrato dai risultati sperimentali, tale approccio è stato successivamente generalizzato per poter permettere lo studio di *dataset* differenti.

I dati ambientali considerati in questo studio si riferiscono ai segnali di temperatura misurati in una scuola elementare per un periodo di 126 giorni (18 settimane) dal 8 dicembre 2011 al 26 marzo 2012. Questi dati hanno un tempo di campionamento di 5 minuti e sono stati

ottenuti attraverso una rete di sensori *wireless* posti in aule adibite ad usi differenti, come la mensa, gli uffici amministrativi, una sala comune, e l'*auditorium*. La seguente immagine mostra la piantina della scuola elementare Rigoni e la locazione di tutti i sensori installati all'interno delle sue stanze.

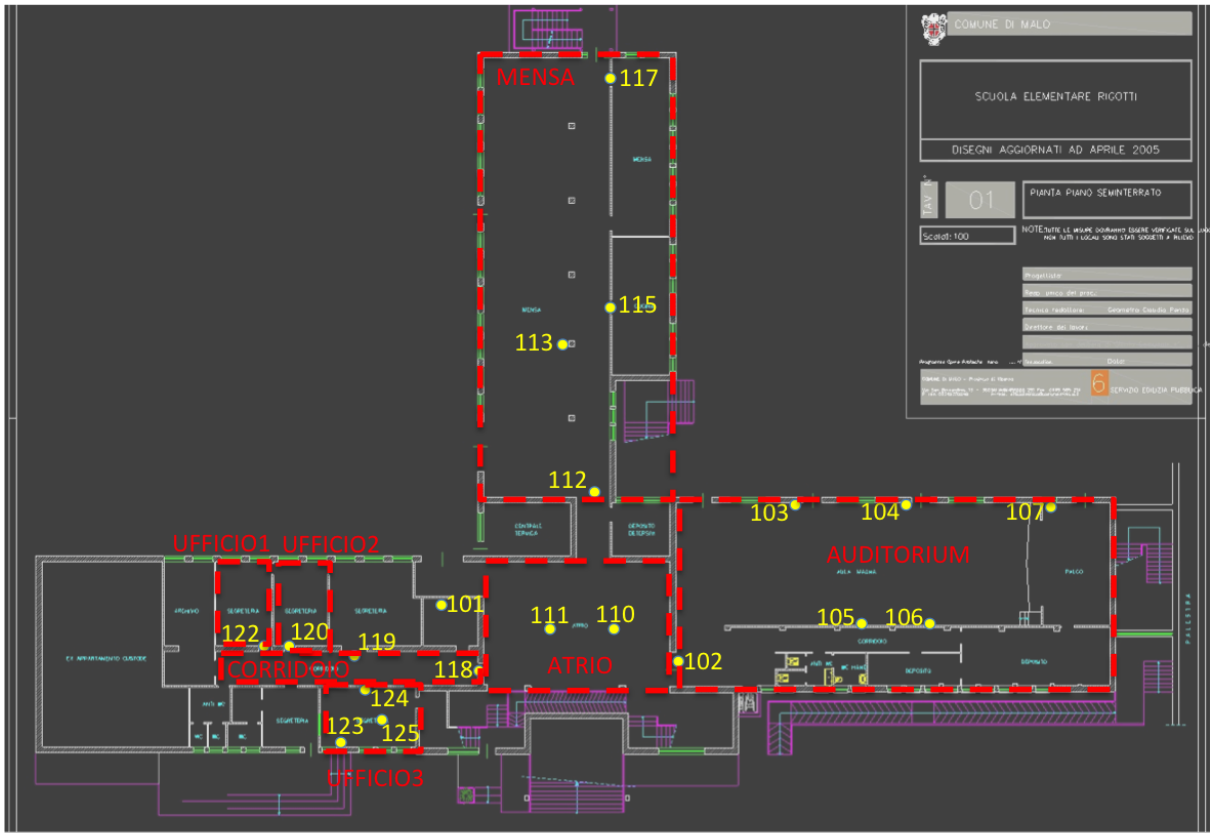

Figura 1.1: Piantina della scuola elementare Rigotti rappresentante la distribuzione dei sensori necessari alla rilevazione delle temperature all'interno delle aule.

Anche se l'applicazione che ha motivato questo lavoro si basa su serie temporali a valori reali, verrà proposta una soluzione che si avvale di una rappresentazione simbolica del segnale di ingresso, in modo da poter operare su stringhe di simboli provenienti da un alfabeto. Nel contesto di rilevamento delle anomalie su sequenze discrete esistono in letteratura tre formulazioni principali per il problema considerato [2]:

1. ***Sequence-based***: individuazione della sequenza più anomala (s) all'interno di un dato insieme di sequenze.
2. ***Subsequence-based***: data una sequenza s, trovare la sua sottosequenza di lunghezza w più anomala.
3. ***Pattern frequency-based***: un determinato *pattern* è considerato anomalo se la sua frequenza all'interno di una sequenza s è inferiore rispetto a quella dello stesso *pattern* rispetto ad una sequenza di allenamento T.

Tra le tecniche che si occupano di analisi basata su un'unica sequenza (formulazione 1), in [3] viene proposta una soluzione basata su *clustering* partizionale nella quale le sequenze di *training* vengono raggruppate attraverso l'algoritmo k-medoids e la distanza LCS (Longest common subsequence). Il grado di anomalia della sequenza da analizzare viene, quindi, calcolato come l'inverso della somiglianza al suo *medoid* più vicino. Tra le tecniche che invece si occupano di analisi basata su sottosequenze (formulazione 2), di particolare interesse sono gli studi in [23] e [24], entrambi basati sulla rappresentazione SAX [7], il metodo di discretizzazione utilizzato per questa tesi; essi verranno discussi nel dettaglio nel capitolo 3.

Il problema qui trattato può essere visto come l'intersezione tra le formulazioni 1 e 2. L'interesse, infatti, è focalizzato sull'individuazione di anomalie di lunghezza specifica, che corrisponde, per esempio, alla durata di un giorno o di una settimana. In tale contesto non è interessante l'analisi di tutte le sottosequenze contenute nel segnale di ingresso, ma bensì la ricerca di anomalie sul suo sottoinsieme di sottosequenze adiacenti.

L'approccio sviluppato in questa tesi, implementato sotto forma di applicativo C++ e denominato `CADET` (Consensus based Anomaly DEtection in Time series), si basa sull'idea di generare un consenso tra tutte le anomalie rilevate da più modelli costruiti mediante un algoritmo di *clustering* applicato al ricampionamento iterato dell'insieme di sottosequenze adiacenti estratte dal segnale di ingresso. Il consenso è rappresentato da un unico modello finale, ottenuto tramite un sistema di voto attuato dalle anomalie rilevate in ciascuna iterazione dell'algoritmo di *clustering* sopra citato. Tale modello potrà essere successivamente riutilizzato per l'analisi di serie temporali differenti da quella di *input*, provenienti dallo stesso sensore di allenamento o da altri sensori installati nel medesimo edificio. Nel caso di ambienti particolarmente rumorosi, si può utilizzare anche la combinazione di modelli provenienti da sensori differenti e più affidabili per costruire una rappresentazione più robusta delle sottosequenze rappresentanti il comportamento "normale" dell'andamento del segnale in esame. Nel contesto specifico che ha motivato questo studio, è possibile avvalersi del supporto di informazioni esterne, come ad esempio un calendario, che indicano quando la scuola elementare è aperta o meno. Questo permetterà di definire come anomali anche quei giorni che possiedono un comportamento "normale" in date in cui la struttura si presuppone chiusa. Il calendario consentirà anche di discriminare tra allarmi e avvisi a seconda se l'anomalia si verifica in una giornata di lavoro o durante una festa.

Il resto della tesi è organizzato come segue: Nel capitolo 2 vengono brevemente definiti i concetti fondamentali sulla teoria delle serie temporali e viene descritta la tecnica di discretizzazione SAX utilizzata da `CADET` per convertire i dati originali da analizzare. Nel capitolo 3 viene presentato e discusso l'algoritmo studiato al fine di rilevare le anomalie su serie temporali basandosi sulla loro rappresentazione simbolica. I risultati sperimentali verranno mostrati ed analizzati nel capitolo 4 mentre, infine, il capitolo 5 riporta le conclusioni e la proposta di lavori futuri.

Capitolo 2

Serie temporali e rappresentazione dei dati

In questo capitolo verranno introdotte le notazioni basilari relative all'analisi di serie temporali e all'utilizzo della tecnica di discretizzazione SAX. Una serie temporale, o serie storica, è una raccolta di osservazioni di dati ben definiti ottenuti attraverso ripetute misurazioni nel tempo. Ad esempio, la rilevazione del valore delle temperature all'interno di una stanza definisce una serie temporale; questo perché la temperatura è un valore ben definibile e costantemente misurabile a intervalli equidistanti. I dati raccolti irregolarmente o sporadicamente non fanno parte di questa definizione. Una serie storica osservata può essere scomposta in tre componenti principali:

trend (direzione a lungo termine): è la tendenza di fondo del fenomeno considerato, riferita ad un lungo periodo di tempo. Essa, in genere rappresentabile mediante una funzione matematica, è utile per capire quale sia l'evoluzione generale di una serie temporale (crescente, calante, stazionaria, oscillatoria).

stagionalità (movimenti legati ad un calendario): è costituita dai movimenti del fenomeno nel corso dell'anno che, per effetto dell'influenza di fattori climatici e sociali, tendono a ripetersi in maniera pressoché analoga nel medesimo periodo. Nel caso di dati relativi ai sensori termici presenti all'interno di una scuola, ad esempio, ci si aspetta che nei periodi feriali e festivi essi siano caratterizzati da andamenti differenti.

irregolarità (fluttuazioni a breve termine): variazioni comportamentali della serie dovute ad eventi casuali od occasionali. La rilevazione di tali sottosequenze indesiderate costituisce l'obiettivo principale dello studio svolto in questo lavoro di tesi.

Le seguenti definizioni formalizzano le notazioni sule quali si basa lo studio dei capitoli successivi.

Definizione 1
Una serie temporale di lunghezza n è una sequenza ordinata di n valori: $T = t_1, ..., t_n$.

Nelle serie che descrivono dati ambientali, come quelli considerati in questo lavoro, i valori sono generalmente numeri reali; nel caso in cui queste sequenze vengano discretizzate (come spiegati più avanti) si parlerà di stringhe di caratteri appartenenti ad un alfabeto Σ.

Definizione 2
Data una sequenza $T = t_1, ..., t_n$ di lunghezza n, la sottosequenza (sottostringa) di lunghezza m avente p come indice iniziale è la sequenza ordinata di posizioni contigue in T: $t_p, ..., t_{p+m-1}$

$(1 \leq p \leq n - m + 1)$.

Nell'analisi delle serie temporali, operare direttamente sui dati grezzi può rivelarsi estremamente oneroso in termini computazionali e di memorizzazione. Pertanto uno dei compiti più ardui nella gestione di serie storiche è quello di scegliere la giusta rappresentazione dei dati; tale decisione influenzerà notevolmente la semplicità e l'efficienza dell'intero processo di *data mining*. Negli ultimi anni sono state proposte in letteratura un numero sempre crescente di rappresentazioni, come ad esempio la famosa trasformata di Fourier, strutture ad albero, modelli polinomiali a tratti, *wavelets*, *eigenwaves* e molte altre. Quasi tutte cercano di descrivere una serie di dati come combinazione di funzioni lineari semplici o periodiche. Senza alcuna pretesa di essere esaustiva, la seguente lista riassume alcune delle rappresentazioni più conosciute ed utilizzate:

- **Data Adaptive**
  - Piecewise Linear Approximation [4]
    - Interpolation
    - Regression
  - Adaptive Piecewise Constant Approximation (APCA) [5]
  - Singular Value Decomposition (SVD) [6]
  - Symbolic
    - Natural Language
    - Strings
      - Non-Lower Bounding
      - Symbolic Aggregate approXimation (SAX) [7]
      - Clipped Data
  - Trees
- **Non-Data Adaptive**
  - Discrete Wavelet Transform (DWT) [8]
  - Random Mappings
  - Spectral
    - Discrete Fourier Transform (DFT) [9]
    - Discrete Cosine Transform (DCT) [10]
    - Chebyshev Polynomials [11]
  - Piecewise Aggregate Approximation (PAA) [12]

Un'importante caratteristica che limita notevolmente gli algoritmi e le strutture dati applicabili su quasi tutte le rappresentazioni appena citate, è che queste sono definite su valori reali; per esempio in alcune applicazioni essere in grado di calcolare la probabilità che una particolare sequenza di valori si verifichi potrebbe aiutare la comprensione di un particolare fenomeno, ma è risaputo che la probabilità di osservare un numero reale è sempre pari a zero. Un altro difetto che colpisce molte rappresentazioni è che queste non concedono una riduzione sostanziale della dimensione dei dati, gravando sulle prestazioni degli algoritmi di *data mining* esistenti che dipendono fortemente dalla lunghezza dell'*input*.

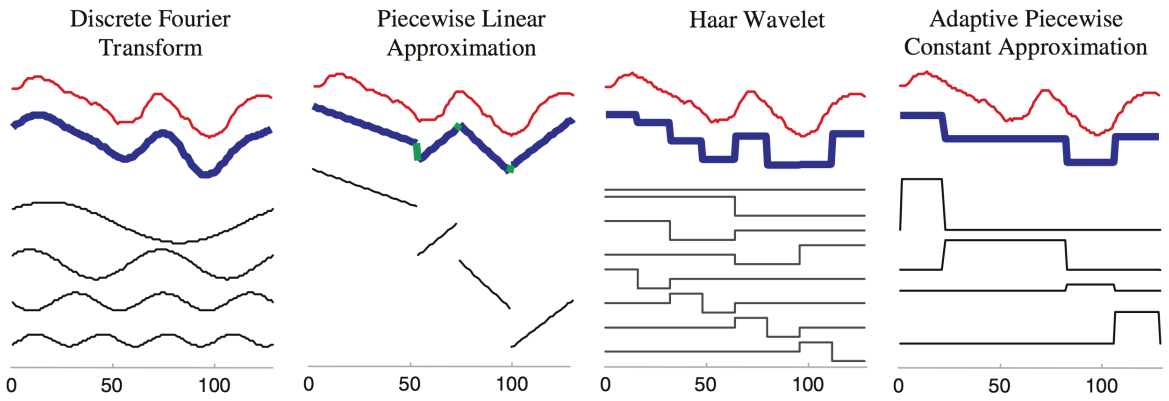

Figura 2.1: Rappresentazioni di serie temporali più utilizzate. Ciascuna di esse cerca di modellare i dati come combinazione lineare di funzioni semplici. Ad esempio la trasformata di Fourier scompone il segnale originale in funzioni periodiche mentre la *Adaptive Piecewise Constant Approximation* in funzioni costanti a tratti. Figura tratta da [7].

Trattare i dati sotto forma di stringa porterebbe a non essere influenzati dai problemi sopra riportati e, vista l'abbondanza di algoritmi e strutture progettate per manipolare le stringhe, l'analisi può essere incentrata su questo tipo di discretizzazione dei dati originali, in particolare sull'approccio SAX. Un grande vantaggio di questa rappresentazione è la possibilità di ridurre estremamente la quantità di dati senza che questi perdano le proprie caratteristiche e le informazioni necessarie per una corretta analisi. Si vedrà in seguito come SAX permetta la definizione di misure di distanza che non sono mai superiori a quelle ottenibili operando direttamente sui dati originali, ed il perché questa proprietà sia fondamentale per ogni algoritmo di *knowledge discovery*.

2.1 SAX: Symbolic Aggregate approXimation

Mediante SAX è possibile ridurre una serie temporale a valori reali di lunghezza n in una stringa di simboli lunga w (tipicamente $w << n$). Ciascun carattere può essere scelto da un alfabeto Σ di grandezza a, specificabile dall'utente (ovviamente $a \geq 2$). La scelta di w e di a dipende dai requisiti applicativi e, poiché condizionerà fortemente i risultati finali, richiede particolare attenzione. Il processo di discretizzazione può essere suddiviso in due fasi principali: durante il primo *step* la serie temporale originale viene trasformata nella rappresentazione PAA, descritta in seguito, la quale subisce un ulteriore conversione in stringa durante il secondo passo dell'algoritmo. La prossima sezione descrive nel dettaglio ciascuna fase richiesta per ottenere la stringa discreta rappresentante i dati originali.

2.1.1 Normalizzazione

La figura 2.2 mostra due serie temporali che hanno una forma molto simile nonostante i loro valori siano diversi per via della scala di valori che utilizzano; mantenere i dati in questa forma rende poco interessante il confronto tra i due andamenti.

Confrontare serie temporali con *offset* ed ampiezza differenti acquista significato solo se esse subiscono prima un processo di normalizzazione, atto a renderle di uguale valore medio e deviazione *standard*. La ***Z-normalization*** richiede in *input* un vettore $\mathbf{x}$ e lo trasforma nell'*array*

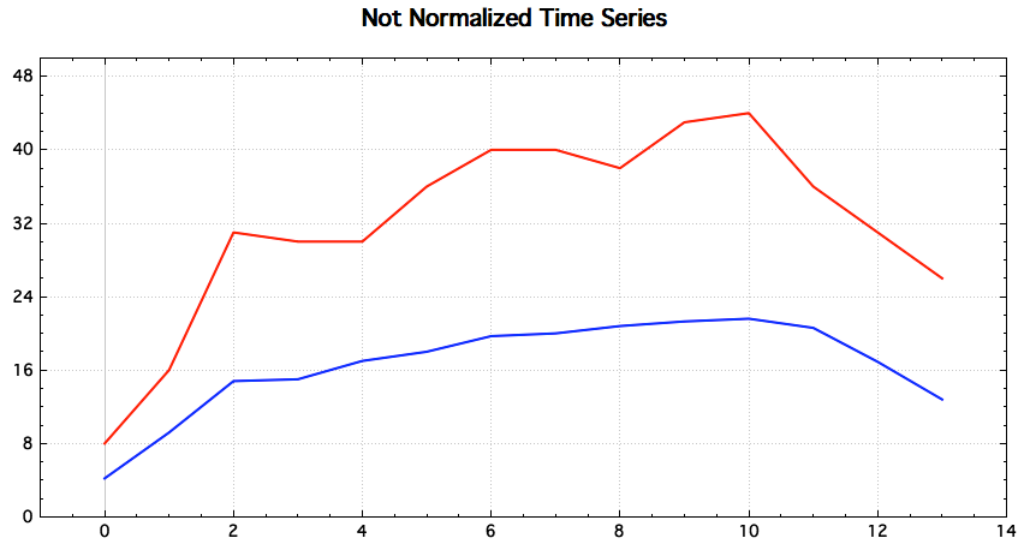

Figura 2.2: Due serie temporali con andamenti molto simili, ma difficili da confrontare per via dei valori molto diversi che le compongono.

$\mathbf{x}'$, dove la media calcolata sugli elementi che compongono l'*output* è approssimativamente zero mentre la deviazione *standard* (e quindi anche la varianza) vale uno.
Questa trasformazione richiede due operazioni:

1. La media μ della serie temporale viene sottratta da ciascuno degli elementi x_i dell'*array* di *input*.

2. Ogni valore ottenuto al punto precedente viene diviso per la deviazione *standard* σ della serie temporale iniziale.

La seguente formula riassume l'intera procedura:

$$x_i^{'} = \frac{x_i - \mu}{\sigma} \quad , \qquad i = 1, ..., n \tag{2.1}$$

Il risultato di tale processo di normalizzazione applicato sulle serie di figura 2.2 è osservabile nell'immagine 2.3.

Nonostante la formula (2.1) appaia estremamente semplice da applicare, esistono due casi ai quali è necessario prestare attenzione. Se una sequenza è quasi sempre costante lungo l'asse dei tempi con qualche sporadico campione affetto da rumore, la normalizzazione tende ad amplificare tali disturbi, il valore dei quali rischia di esplodere ad ampiezze molto elevate. Fortunatamente è facile accorgersi di questa complicazione poiché la deviazione *standard* di un segnale così formato è molto bassa; per aggirare tale situazione, che occorre più frequentemente di quanto si possa immaginare, si può assegnare a tutti gli elementi della serie normalizzata uno stesso valore (piccolo), per esempio 0.1. Un caso molto più raro, ma che bisogna prevedere e gestire, capita quando la serie temporale è formata da un unico valore, compromettendo il calcolo della deviazione *standard* che non può essere definita. Anche qui la soluzione non è complicata, poiché basta generare in *output* un vettore contenente un singolo valore unitario.

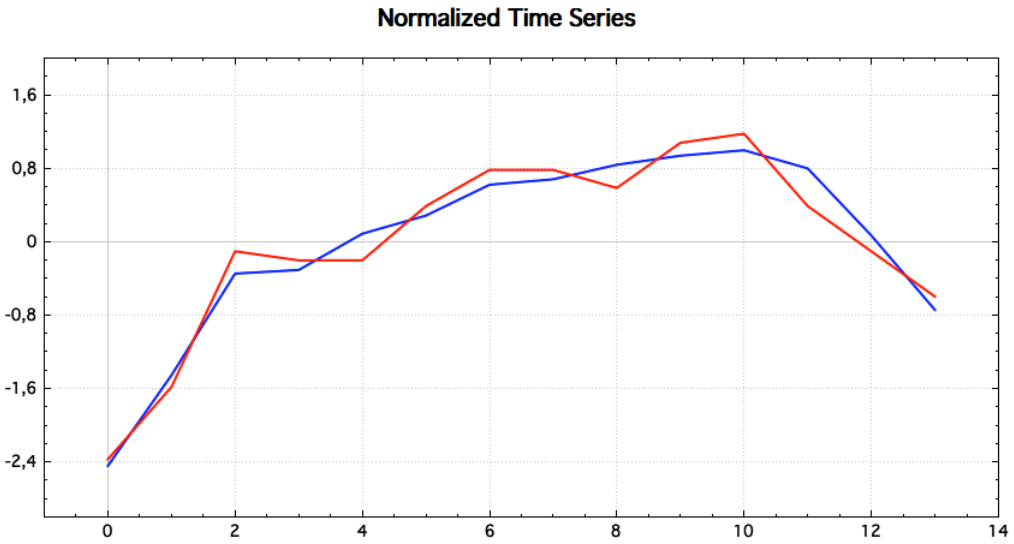

Figura 2.3: Le stesse serie temporali mostrate in figura 2.2, ma entrambe normalizzate e quindi facilmente confrontabili.

2.1.2 Riduzione della dimensionalità attraverso la trasformazione PAA

Come anticipato precedentemente, per rappresentare una serie temporale attraverso SAX, dopo essere stata normalizzata questa deve subire un processo di approssimazione mediante la trasformata PAA. Tale algoritmo converte una serie temporale $T = t_1, t_2, ..., t_n$ in una serie $\overline{T}$ di lunghezza w, dove solitamente $w << n$. $\overline{T}$ è calcolata dividendo T in w segmenti della medesima dimensione (chiamati ***frame***), ciascuno dei quali viene mappato in un elemento della nuova sequenza mediante la seguente equazione:

$$\overline{t_i} = \frac{w}{n} \cdot \sum_{j=\frac{n}{w}(i-1)+1}^{\frac{n}{w}i} t_j \tag{2.2}$$

Sostanzialmente, vengono calcolati i valori medi dei dati appartenenti a ciascun *frame* ed un vettore contenente tali medie diventa la rappresentazione ridotta dei dati di partenza, come mostrato in figura 2.4. Nonostante tale procedura appaia molto semplice, essa è in grado di competere con alcune delle tecniche più sofisticate presentate all'inizio di questo capitolo.

Una diretta conseguenza dell'algoritmo sopra descritto è che esso crea *frame* di dimensione $\frac{n}{w}$. È fondamentale, dunque, gestire la situazione in cui n non è divisibile per w. Per esempio, se si decide di suddividere una serie di 100 campioni in 3 segmenti, quello che si ottiene è un *frame size* di $33.\overline{3}$ e ci saranno necessariamente alcuni punti dei quali non è ben definita l'appartenenza ad una finestra temporale precisa. Esistono molte alternative per aggirare tale mancanza di univocità:

- La prima soluzione brutale è quella di eliminare tutti i punti in eccesso, con una conseguente perdita di informazione. Nell'esempio precedente solo un punto verrebbe rimosso, ma se si decidesse di suddividere il segnale in 15 *frame* il numero di campioni esclusi diventerebbe 10. Tale procedura, però, non è sempre utilizzabile in quanto esistono molte applicazioni nelle quali non è ammesso ridurre la quantità di informazione iniziale.

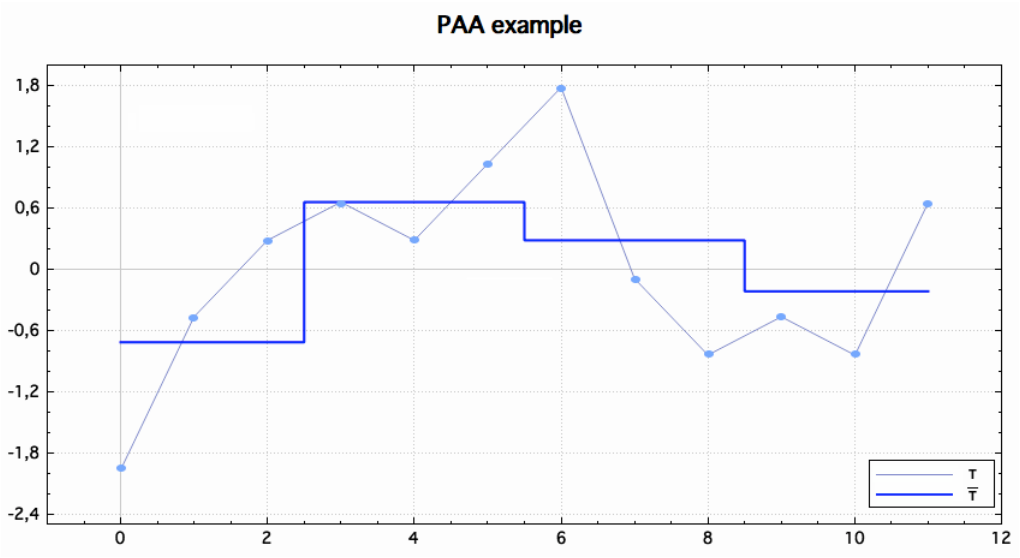

Figura 2.4: Una serie temporale di lunghezza 12 ridotta a 4 valori attraverso la trasformazione PAA.

- Se l'applicazione lo consente, è possibile rilassare il vincolo di avere tutti i *frame* della stessa dimensione in modo da ridistribuire i punti che sarebbero stati rimossi dall'approccio precedente, mantenendo tutte le informazioni contenute nella sequenza originale. Applicando tale procedura all'esempio sopra descritto si potrebbero creare due segmenti di dimensione 33 ed uno di lunghezza 34, senza perdere alcun campione.
- Una soluzione più accurata prevede di assegnare un peso relativo al contributo su un *frame* di ciascun punto della serie temporale. Per ogni campione, invece che inserirlo interamente in un segmento, è possibile associarne solo una parte proporzionale al suo peso. Per esempio una serie temporale di lunghezza 10 campioni può essere suddivisa in *frame* di dimensione 3.3 punti, pesando quest'ultimi come mostrato in figura 2.5. Tale approccio è quello utilizzato in questo lavoro di tesi.

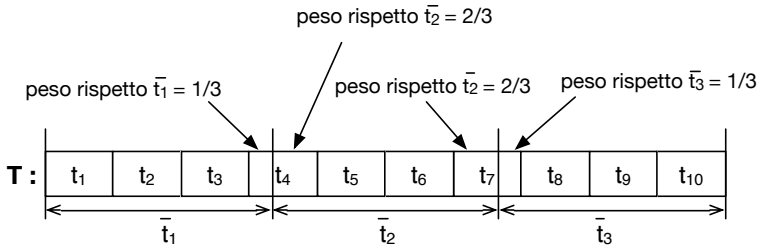

Figura 2.5: Dieci punti suddivisi in 3 segmenti pensando i campioni di frontiera. Per i punti il cui peso non è indicato, esso si intende pari a uno.

2.1.3 Discretizzazione

Nell'ultima fase dell'approccio SAX, la rappresentazione PAA viene sostituita da una stringa di simboli, ciascuno dei quali è selezionato da un alfabeto $\Sigma = \{\alpha_1, \alpha_2, ..., \alpha_a\}$[1]. Per riflettere

[1]In questa tesi vengono utilizzate le prime 20 lettere dell'alfabeto latino classico, $\alpha_1 = \mathbf{a}$, $\alpha_2 = \mathbf{b}$ e così via.

al meglio le caratteristiche della serie temporale, una discretizzazione ideale dovrebbe produrre simboli equiprobabili [13]; tale risultato è ottenibile grazie al fatto che quasi tutti i segnali normalizzati possiedono una distribuzione Gaussiana. Per il limitato insieme di casi che non ubbidiscono a tale previsione il rendimento del processo di *data mining* potrebbe subire un leggero peggioramento, ma la correttezza degli algoritmi applicabili non risulterebbe compromessa [7]. Al fine di utilizzare tale risultato per la discretizzazione mediante SAX, l'area sottostante la distribuzione normale (di media 0 e varianza 1) viene suddivisa in a regioni delimitate dai ***breakpoint*** $B = \beta_0, \beta_1, ..., \beta_a - 1, \beta_a$. L'area di N(0, 1) compresa tra β_i e β_{i+1} deve valere necessariamente $\frac{1}{a}$ ($\beta_0 = -\infty$, $\beta_a = \infty$). Tutti i coefficienti della rappresentazione PAA il cui valore è inferiore a β_1 vengono mappati sul simbolo $\alpha_1 = \mathbf{a}$, tutti quelli più grandi o uguali a β_1 e minori di β_2 sono rappresentati da $\alpha_2 = \mathbf{b}$, etc. La concatenazione di w simboli risultante viene chiamata **parola** ed è indicata come $\widehat{T}$.

Osservando che i *breakpoint* non dipendono dai dati di *input* ma solo dalla taglia dell'alfabeto, è possibile costruire una *lookup table* di *breakpoint* per ciascun valore assegnabile ad a. La tabella 2.1 mostra i *breakpoint* per una taglia dell'alfabeto che varia tra 2 e 10 simboli.

	2	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
$\beta_\mathbf{1}$	0	-0.43	-0.67	-0.84	-0.97	-1.07	-1.15	-1.22	-1.28
$\beta_\mathbf{2}$	-	0.43	0	-0.25	-0.43	-0.57	-0.67	-0.76	-0.84
$\beta_\mathbf{3}$	-	-	0.67	0.25	0	-0.18	-0.32	-0.43	-0.52
$\beta_\mathbf{4}$	-	-	-	0.84	0.43	0.18	0	-0.14	-0.25
$\beta_\mathbf{5}$	-	-	-	-	0.97	0.57	0.32	0.14	0
$\beta_\mathbf{6}$	-	-	-	-	-	1.07	0.67	0.43	0.25
$\beta_\mathbf{7}$	-	-	-	-	-	-	1.15	0.76	0.52
$\beta_\mathbf{8}$	-	-	-	-	-	-	-	1.22	0.84
$\beta_\mathbf{9}$	-	-	-	-	-	-	-	-	1.28

Tabella 2.1: *Lookup table* contenente i *breakpoint* per una taglia dell'alfabeto da 2 a 10 caratteri.

Utilizzando tale tabella è possibile convertire tutti i valori reali della rappresentazione PAA nei corrispondenti simboli di Σ come segue:

$$\widehat{t_i} = \alpha_j \quad iif \quad \beta_{j-1} \leq \bar{t}_i < \beta_j \tag{2.3}$$

La figura 2.6 mostra la rappresentazione SAX della serie temporale dell'immagine 2.4 usando una taglia dell'alfabeto pari a 5.

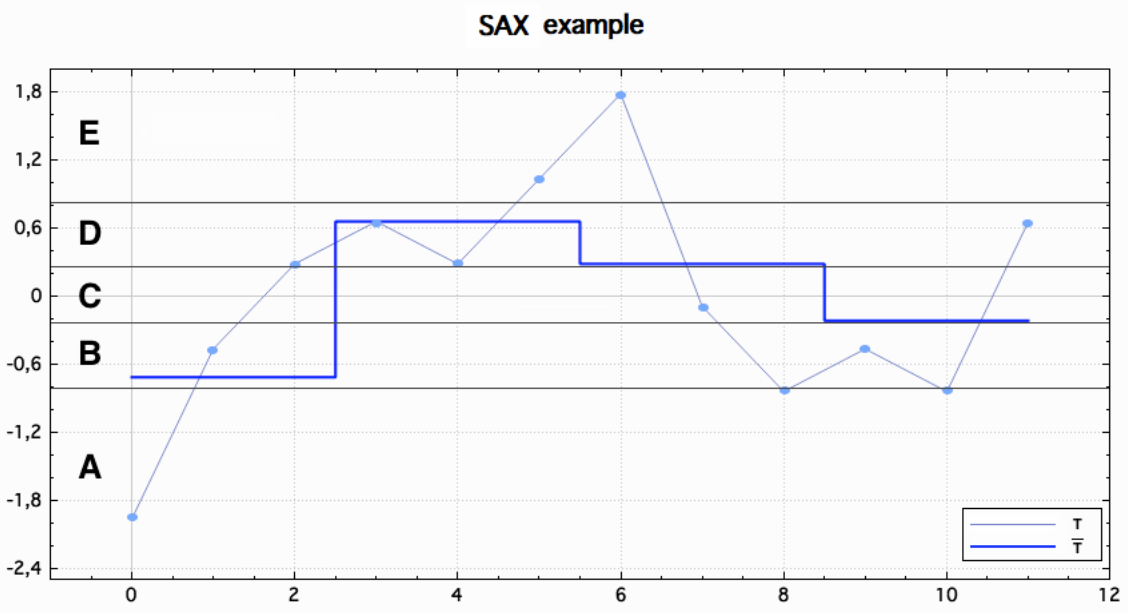

Figura 2.6: Discretizzazione SAX della serie temporale di figura 2.4 utilizzando *breakpoint* precalcolati (tabella 2.1). Nell'esempio, con $n = 12$, $w = 4$ e $a = 5$, la parola risultante è "**bddc**".

2.2 Misure di distanza

Allo scopo di confrontare due serie temporali differenti, è necessario determinare un modo per calcolare la distanza tra di esse; più bassa sarà la distanza e più le serie risulteranno simili. SAX introduce una nuova metrica per misurare la somiglianza tra stringhe estendendo la distanza Euclidea e quella basata su PAA.

La **distanza Euclidea** tra due serie temporali $T = \{t_1, ..., t_n\}$ e $C = \{c_1, ..., c_n\}$ della stessa lunghezza è definita dalla seguente equazione:

$$\text{Eucl Dist(T, C)} = \sqrt{\sum_{i=1}^{n}(t_i - c_i)^2} \tag{2.4}$$

Indicando con $\overline{T} = \{\bar{t}_1, ..., \bar{t}_w\}$ e $\overline{C} = \{\bar{c}_1, ..., \bar{c}_w\}$ le rappresentazioni di T e C convertite in PAA, l'equazione 2.4 può essere facilmente modificata per ottenere una approssimazione che ammette il *lower bounding*[2] della distanza euclidea tra le due serie originali:

$$\text{PAA Dist}(\overline{T}, \overline{C}) = \sqrt{\frac{n}{w}} \cdot \sqrt{\sum_{i=1}^{w}(\bar{t}_i - \bar{c}_i)^2} \tag{2.5}$$

Se, infine, i dati vengono trasformati nella rappresentazione SAX, è possibile definire una misura di similarità che restituisca la distanza minima tra due parole:

$$\text{SAX Dist}(\widehat{T}, \widehat{C}) = \sqrt{\frac{n}{w}} \cdot \sqrt{\sum_{i=1}^{w}\left(\text{dist}(\widehat{t}_i, \widehat{c}_i)\right)^2} \tag{2.6}$$

[2]Con il termine *Lower bounding* si intende PAA Dist(T, C) $\leq$ Eucl Dist(T, C) $\forall$ T, C.

La funzione interna *dist()*, che calcola la distanza minima tra due simboli appartenenti all'alfabeto, può essere implementata attraverso una *lookup table* simile a quella di tabella 2.2. Per ciascun valore della taglia dell'alfabeto a tale tabella necessita di essere generata una sola volta.

	1 (a)	**2 (b)**	**3 (c)**	**4 (d)**	**5 (e)**
1 (a)	0	0	0.59	1.09	1.68
2 (b)	0	0	0	0.5	1.09
3 (c)	0.59	0	0	0	0.59
4 (d)	1.09	0.5	0	0	0
5 (e)	1.68	1.09	0.59	0	0

Tabella 2.2: *Lookup table* utilizzata dalla funzione di distanza di SAX per una taglia dell'alfabeto pari a 5. La distanza tra due simboli può essere letta in tempo costante alla corrispondente riga e colonna. Per esempio, la distanza tra i caratteri 'a' e 'd' è pari a 1.09.

Ciascuna cella della tabella *dist* dipende solo dai valori dei *breakpoint* descritti in sezione 2.1.3 è può essere calcolata come segue:

$$\text{dist}(i,j) = \begin{cases} 0 & \text{if } \ |i-j| \leq 1 \\ \beta_{\max(i,j)-1} - \beta_{\min(i,j)} & \text{otherwise} \end{cases} \tag{2.7}$$

Per esempio, utilizzando una taglia dell'alfabeto di 5, la distanza tra i simboli 'b' e 'd' è memorizzata alla cella (2, 4). Osservando la tabella 2.1, i *breakpoint* per questo particolare valore di a sono $\beta_1 = -0.84$, $\beta_2 = -0.25$, $\beta_3 = 0.25$ e $\beta_4 = 0.84$, quindi $dist(b,d) = \beta_{\max(2,4)-1} - \beta_{\min(2,4)} = \beta_3 - \beta_2 = 0.5$.

Come introdotto precedentemente e dimostrato in [12] e [7], il grande vantaggio della funzione di similarità definita su SAX è la proprietà di *lower bounding* della distanza Euclidea. In pratica, la distanza calcolata sui dati approssimati mediante l'equazione (2.6) è sempre inferiore a quella calcolabile sulle serie originali attraverso la (2.4). Questa caratteristica è indispensabile, poiché permette di applicare molti degli algoritmi di *data mining* esistenti sulle rappresentazioni SAX producendo gli stessi risultati che si otterrebbero sui segnali di partenza. Per esempio, un'anomalia rilevata utilizzando la rappresentazione SAX e la formula (2.6) è sicuramente tale anche per i dati originali, in quanto la sua distanza da un certo comportamento ritenuto "normale" è sicuramente inferiore di quella reale (nel significato dell'equazione 2.4). Pensando ad una rilevazione basata su *threshold*, se una sequenza discretizzata non supera tale soglia, non la potrà superare nemmeno la corrispondente rappresentazione di partenza.

C'è un ultima osservazione sull'utilizzo dell'equazione 2.6: poiché il valore ritornato come misura di distanza tra due sequenze dipende dalla taglia dell'alfabeto a e dalle dimensioni n e w, non è possibile predire in anticipo quale sarà il più alto valore ammissibile come similarità tra due serie temporali. Pertanto è impossibile applicare algoritmi sui dati che prevedano di fissare un valore di soglia statico sulla lontananza di due stringhe. In generale questo non è un problema irrisolvibile, nel prossimo capitolo si vedrà come gestire gli algoritmi che necessitano

di tale requisito.

La figura 2.7 mostra una rappresentazione visiva delle tre misure di distanza presentate in questa sezione.

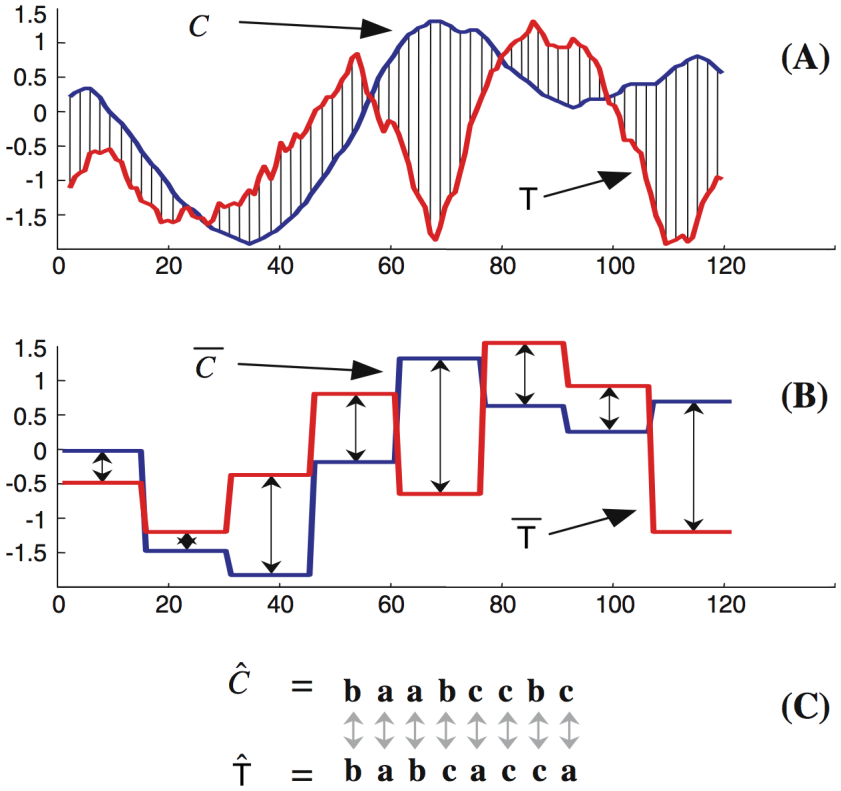

Figura 2.7: Una rappresentazione visuale delle tre misure di distanza presentate in questa sezione. (**A**) La distanza Euclidea tra due serie temporali esamina ogni coppia di punti corrispondenti. (**B**) La distanza definita su rappresentazioni PAA esamina ogni coppia di coefficienti, rappresentati nel grafico come segmenti. (**C**) La distanza tra due parole SAX richiede il calcolo della similarità tra ogni coppia di simboli. Figura tratta da [7].

2.3 La rappresentazione iSAX

Il sempre crescente interesse della comunità scientifica nei confronti di SAX ha aperto le porte verso nuovi studi e verso nuove necessità, tra le quali quella di avere una rappresentazione capace di indicizzare e successivamente analizzare in maniera efficiente database di dimensioni estremamente elevate (sull'ordine del Terabyte). iSAX (***i**ndexable **S**ymbolic **A**ggregate approX**i**mation*) è la più famosa tra le estensioni all'approccio classico di SAX precedentemente discusso [14]. Tale tecnica si basa su alcune modifiche atte a rendere SAX una rappresentazione multi-risoluzione al fine di permettere l'indicizzazione dei dati mediante *hashing* estendibile. Utilizzando la stessa notazione vista in questo capitolo, una serie temporale T può essere rappresentata come:

$$\text{SAX}(T, w, a) = T^a = \left\{t_1^a, t_2^a, ..., t_{w-1}^a, t_w^a\right\}$$

dove i singoli valori t_i, anziché essere rappresentati da interi o caratteri, sono sostituiti da numeri binari. Il numero b di bit necessari per rappresentare ciascun t_i viene chiamato **livello di**

risoluzione ($b = \lceil log_2\ a \rceil$). La seguente figura mostra come sia possibile convertire una serie temporale utilizzando diversi livelli di risoluzione.

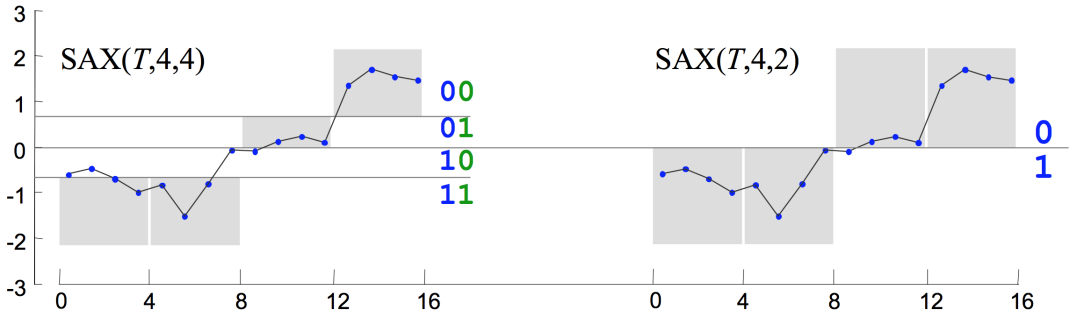

Figura 2.8: Una serie temporale di lunghezza 16 convertita in SAX utilizzando due livelli di risoluzione (**sinistra**) ed un solo livello (**destra**). Figura tratta da [14].

Una rappresentazione multi-risoluzione, per poter funzionare, deve necessariamente prevedere dei meccanismi di confronto tra parole di cardinalità (intesa come taglia dell'alfabeto scelto) diverse. La maniera più semplice per ottenere due parole confrontabili è quello di convertirne le cardinalità in modo che siano uguali e successivamente applicare l'equazione 2.6. Per raggiungere tale obiettivo è possibile percorrere due strade differenti:

Retrocessione della parola con cardinalità superiore
È possibile dimezzare la cardinalità di una parola semplicemente eliminando il bit di coda di ciascun simbolo che la compone. Ovviamente tale proprietà è ricorsiva per cui partendo, per esempio, dalla parola $T^{32} = \{10110, 10101, 00011, 01011, 11011\}$ con cardinalità 32 è possibile ottenerne una con un solo livello di risoluzione mediante i seguenti passaggi:

- $T^{32} = \{10110, 10101, 00011, 01011, 11011\}$
- $T^{16} = \{1011, 1010, 0001, 0101, 1101\}$
- $T^{8} = \{101, 101, 000, 010, 110\}$
- $T^{4} = \{10, 10, 00, 01, 11\}$
- $T^{2} = \{1, 1, 0, 0, 1\}$

Ovviamente tale compressione non è reversibile e comporta una perdita di informazione, ma fornisce un *lower bound* ammissibile sulla distanza tra due stringhe [14].

Promozione della parola con cardinalità inferiore
Tale conversione risulta leggermente più complessa della precedente, in quanto l'aggiunta di un nuovo livello di risoluzione non può essere effettuata in maniera arbitraria. Per esempio, si consideri il tentativo di promuovere T^2 dell'esempio precedente in una parola di cardinalità 8 al fine di confrontarla con $C^8 = \{010, 100, 111, 010, 011\}$. Indichiamo la parola cercata con $T^8 = \{1**, 1**, 0**, 0**, 1**\}$ dove gli asterischi rappresentano i bit dei quali vogliamo definire il valore. Il seguente frammento di pseudo-codice illustra come procedere:

```
IF S^k_i forms a prefix for T^8_i THEN,
        *_i = T^8_i for all unknown bits.
ELSE IF S^k_i is lexicographically smaller than
corresponding bits in T^8_i THEN,
        *_i = 1 for all unknown bits.
ELSE,
        *_i = 0 for all unknown bits.
```

La parola ottenuta dopo tale conversione è $T^8 = \{100, 100, 011, 010, 100\}$ che, come ci si aspettava, è diversa da quella da cui si era partiti nell'esempio precedente.

La potenza espressiva di iSAX permette, infine, di rappresentare una parola utilizzando una cardinalità diversa per ciascun simbolo. Per esempio, è possibile ammettere parole del tipo $\{110, 11, 111, 1\} = \left\{6^8, 3^4, 7^8, 1^2\right\}$. Per confrontare parole così formate, è sufficiente allineare ciascun simbolo corrispondente e promuovere il simbolo con cardinalità inferiore. Una soluzione alternativa, raramente utilizzata, consiste nel rendere uguale la risoluzione di tutti i simboli di entrambe le rappresentazioni.

Per i dettagli sulla costruzione dell'indice e sulle tecniche di ricerca (esatta ed approssimata) sui dati indicizzati si rimanda all'articolo originale degli autori di iSAX [14].

2.4 Applicazioni di SAX

Fin dalla sua introduzione, SAX ha incuriosito un numero sempre crescente di ricercatori, con un conseguente impatto nel settore industriale ed accademico. L'interesse nei confronti di questo approccio è stato così grande che molti studi hanno cercato, oltre che a sfruttarne le caratteristiche, di estenderne le potenzialità. La motivazione è molto semplice: gli autori di SAX hanno dimostrato in [7] che la loro rappresentazione è buona almeno quanto le DFT, DWT, SVD, etc. per molti dei classici algoritmi di *data mining* quali classificazione, *clustering* e *indexing*; nel caso di *anomaly detection* e *motif finding* essi affermano che l'approccio SAX attualmente sia quello che permetta di ottenere i risultati migliori. Combinando queste promettenti motivazioni con le proprietà sopra descritte di *lower bounding* e *dimensionality/numerosity reduction* è possibile ottenere una rappresentazione estremamente competitiva. Sfogliando la bibliografia degli studi svolti avvalendosi di SAX, è interessante notare come molti di essi siano ispirati ad algoritmi per la bioinformatica. Per esempio, in [16] gli autori utilizzano SAX e l'algoritmo di *random projection*[3] per la ricerca di motivi in serie temporali telemediche. Senza nessuna pretesa di essere esaustiva, questa sezione presenta alcuni dei lavori più interessanti basati sull'utilizzo di SAX per la rappresentazione dei dati (Una lista più ricca è situata sul sito ufficiale degli autori di SAX).

2.4.1 Data mining su sagome

Si consideri la sagoma di un oggetto come un immagine binaria rappresentante il suo contorno. Definendo un modo per convertire sagome in due dimensioni in una rappresentazione monodimensionale (*pseudo time series*) è possibile applicare SAX con l'obiettivo di eseguire algoritmi

[3]Random Projection è un metodo per convertire dei dati con dimensionalità elevata in rappresentazioni più compatte. Come suggerisce il nome, tale proiezione viene effettuata rimuovendo casualmente alcune dimensioni dai dati, e reiterata per coprire al meglio lo spazio di ricerca.

di *data mining* su di essa. In [15] gli autori introducono il problema della ricerca dei *top-k shape discords*, le sagome più insolite all'interno di una collezione di immagini. Essi argomentano gli effetti di scala, rumore, occlusione, distorsione e rotazione sui dati. Per convertire un contorno in serie temporale viene utilizzato l'approccio basato su distanza dal centroide, un metodo molto efficace per preservare le caratteristiche delle immagini di partenza. In questo tipo di conversione, viene misurata la distanza Euclidea da ciascun punto del profilo al centroide, e quest'ultima viene considerata come il valore sull'asse Y di una serie temporale di lunghezza n. Un applicazione comune di questa tecnica è la ricerca di oggetti dannegiati all'interno di un database di sagome, come mostrato dalla figura sottostante per un insieme di immagini di ali di insetti.

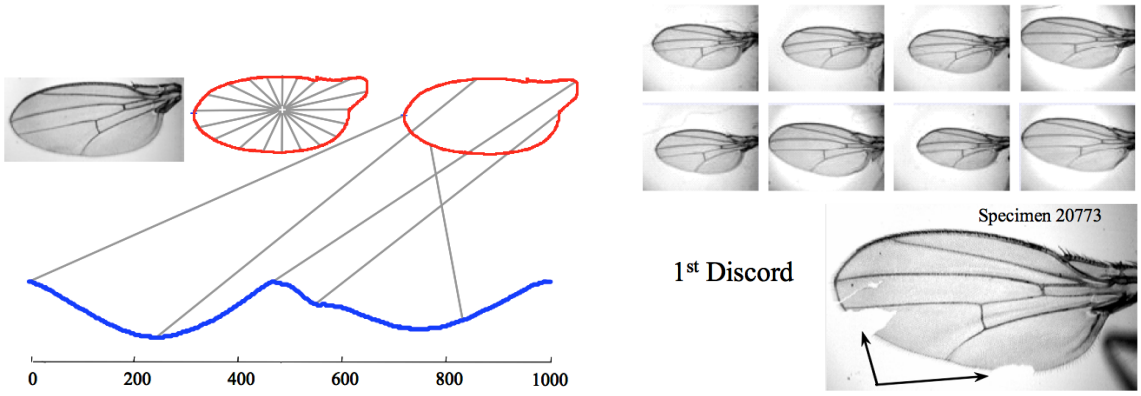

Figura 2.9: **Left**: Conversione di ciascuna ala in serie temporale utilizzando l'approccio basato su distanza dal centroide. textbfRight: La prima sagoma individuata dall'algoritmo è quella di un ala danneggiata. Figura tratta da [15].

Altri studi simili includono problemi di *annotation* (dato un oggetto di interesse, si vogliono ottenere in maniera automatica informazioni aggiuntive su di esso), *query by content* (dato un grande insieme di dati, si desidera individuare i primi k oggetti simili ad uno di interesse), *clustering* e classificazione.

2.4.2 Autenticazione di impronte palmari

L'autenticazione delle persone attraverso l'utilizzo di caratteristiche biometriche (impronta digitale, riconoscimento vocale, scansione della retina, ecc.) è un argomento molto attuale che interessa ogni giorno migliaia di ricercatori nel mondo. Uno studio recente, riguardante l'utilizzo di impronte palmari per il riconoscimento dell'identità di un individuo, ha interessato molti scienziati poiché i dati ottenibili da tali scansioni contengono una quantità elevata di informazione e possono essere rappresentati in moltissime maniere differenti. In [17] gli autori utilizzano SAX per convertire impronte palmari in serie temporali e si avvalgono della distanza (2.6) per calcolare un *matching score* tra due *template* differenti. La figura 2.10 mostra l'accoppiamento di due scansioni estratte da immagini differenti prese dallo stesso individuo.

2.4.3 Gestione di dati metereologici

Molte applicazioni interessanti di SAX riguardano il monitoraggio di dati climatici. In [18] gli autori costruiscono un modello Markoviano spazio-temporale per il controllo dell'evoluzione della copertura della foresta pluviale Brasiliana. Tale monitoraggio è gestito da sensori che rilevano variazioni di temperatura causate da movimenti d'aria, denominati **microfont**. Con l'intento

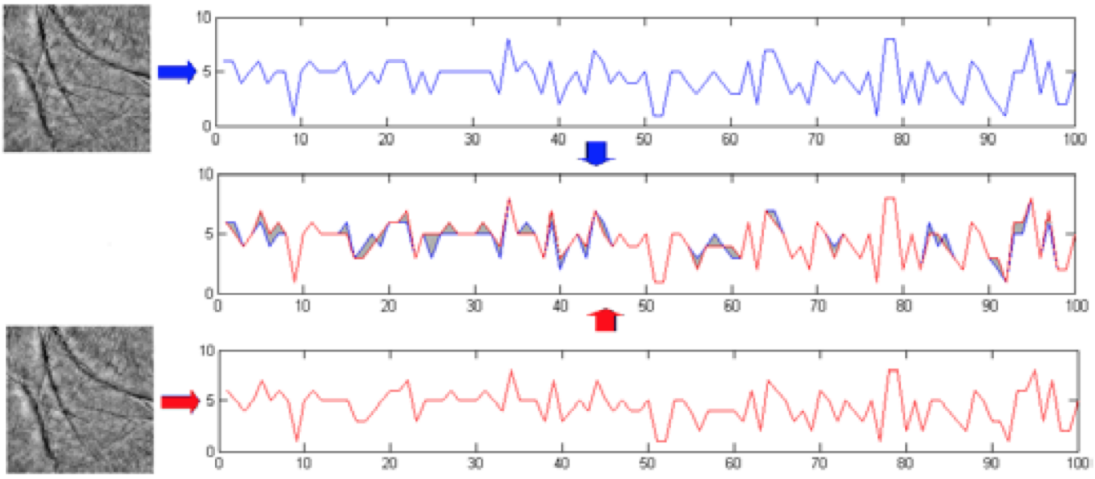

Figura 2.10: Accoppiamento di impronte palmari basato sul calcolo della distanza SAX. Più le aree grigie sono piccole, minore è la distanza tra le due serie. Figura tratta da [17].

di fornire uno studio convincente gli autori discutono, inoltre, il problema della distribuzione intelligente dei sensori al fine di identificare i *microfont* più significativi.

Un altro lavoro svolto nell'ambito del *data mining* di dati meteorologici riguarda la previsione della formazione di tornadi [19]. Invece che analizzare i dati forniti dai sensori *radar*, gli autori basano il loro studio sulla costruzione di una griglia tridimensionale contenente le tre componenti principali del moto dell'aria: temperatura, pressione e precipitazione. SAX è applicato a ciascuna delle tre serie temporali e un algoritmo di *rule finding* è eseguito al fine di comprendere al meglio come riuscire a predire l'evoluzione dei tornadi.

2.4.4 Visualizzazione

L'occhio umano è considerato spesso l'ultimo strumento di *data mining*. L'onnipresente grafico di serie temporali è utilizzato per semplificare la visualizzazione dei dati in molti algoritmi di *knowledge discovery*, tuttavia esistono casi reali nei quali la quantità di dati è massiccia e non è possibile visualizzare l'intera informazione necessaria per colpa delle basse dimensioni concesse dai dispositivi di *output* (schermi, stampanti, ecc.). Un applicazione molto interessante di SAX riguarda la generazione di icone intelligenti (chiamate ***bitmap***) capaci di rappresentare in uno spazio ridotto tutta l'informazione necessaria per il confronto di dati. Osservando le similarità e le differenze tra *bitmap* all'interno di una collezione, l'utente può velocemente scoprire *cluster*, anomalie e altre irregolarità nei dati in essa contenuti. In [21] gli autori definiscono una rappresentazione basata sul famoso algoritmo ***chaos game*** [20]. Dopo aver convertito una serie temporale in una stringa di lunghezza n, l'idea di base è quella di mappare le frequenze di ciascuna sottostringa lunga k su una matrice quadrata di dimensione $2k \times 2k$, e successivamente colorare ogni cella secondo il valore in essa contenuto. Una buona scelta di colorazione può essere quella di normalizzare le frequenza nell'intervallo [0, 255] ed utilizzare tali valori come rappresentazione in scala di grigi. Poiché è noto che la scala di grigi non è percettivamente uniforme, un'alternativa migliore può essere quella di utilizzare lo spazio di colori RGB. Le *bitmap* generate mediante l'algoritmo *chaos game* sono definite solo per sequenze contenenti simboli appartenenti ad un alfabeto di 4 caratteri. Questo può essere limitante per molti algoritmi che richiedono una taglia di Σ superiore, ma molti autori hanno dimostrato che una cardinalità

di quattro è una scelta ottima per svariati *dataset*. La figura 2.11 riassume la procedura di conversione.

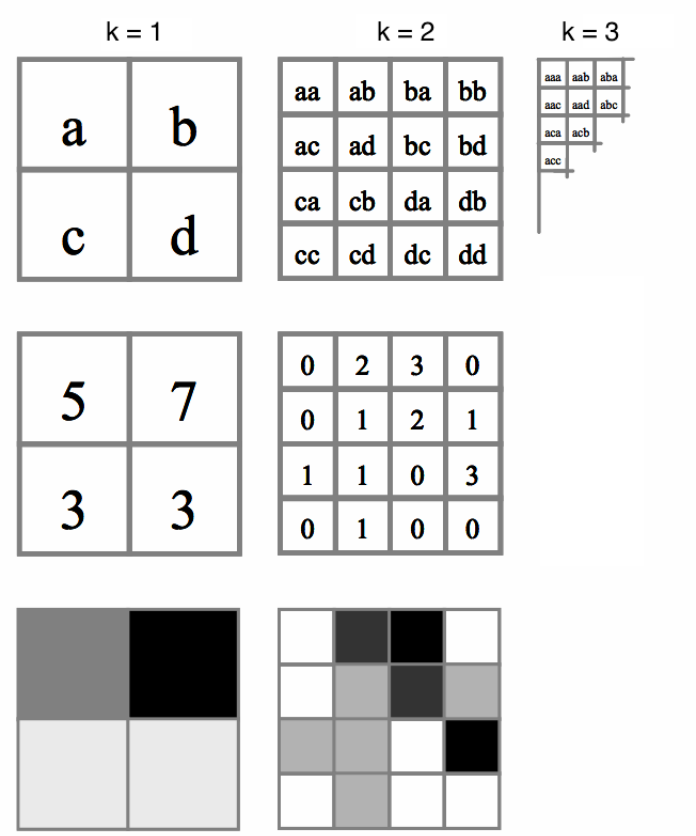

Figura 2.11: Rappresentazione tramite icona di una serie temporale convertita nella rappresentazione SAX con un alfabeto di 4 caratteri. Inizialmente tutti i *k-mer* sono estratti dalla parola, successivamente le frequenze sono calcolate, ed infine tali valori sono convertiti in colore. Figura tratta da [21].

Per *bitmap* della stessa dimensione, è possibile definire una misura di distanza tra queste come la sommatoria dei quadrati delle distanze di ogni coppia di *pixel*. Formalmente, per due icone B_1 e B_2 di dimensione $k \times k$, la distanza tra esse è espressa come:

$$dist(B_1, B_2) = \sum_{i=1}^{k} \sum_{j=1}^{k} (B_1^{ij} - B_2^{ij})^2 \tag{2.8}$$

Una volta definita la distanza tra *bitmap*, è semplice utilizzarla all'interno di un algoritmo di *clustering*, come mostrato in figura 2.12. Nel prossimo capitolo verrà illustrato come utilizzare la rappresentazione appena descritta per effettuare ricerca di anomalie su serie temporali.

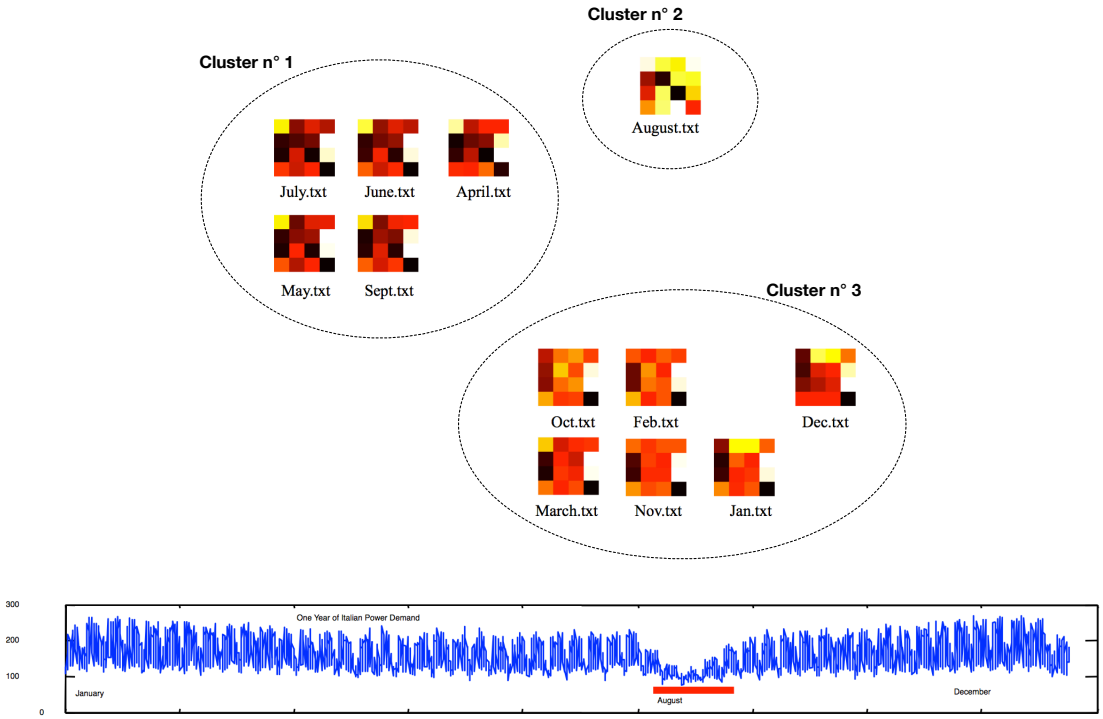

Figura 2.12: Una serie temporale con dati rilevati nell'arco di un anno viene suddivisa in 12 mesi e questi ultimi vengono convertiti in *bitmap* ed utilizzati come *input* di un algoritmo di *clustering*. Figura tratta da [22].

Capitolo 3

Ricerca di anomalie

I recenti avanzamenti tecnologici hanno permesso la costruzione di sensori sempre più veloci e precisi, capaci di raccogliere dati di varia natura in tempo reale. Vista la grande quantità di informazione che questi sono capaci di raccogliere gli studiosi si sono trovati di fronte ad un nuovo problema da risolvere: la ricerca di anomalie sui segnali rilevati. Questa operazione, difficile da effettuare manualmente, risulta estremamente onerosa anche per un algoritmo di ricerca classico, che si vede ad agire su dati il cui volume è insostenibile dalle risorse medie di un elaboratore. Molte soluzioni precedenti prevedevano la costruzione di programmi ad hoc che, naturalmente, richiedevano grande lavoro e permettevano poche possibilità di adattamento in applicazioni diverse da quella per la quale erano stati concepiti. Grazie a tecniche di discretizzazione simbolica dei dati, in particolare con l'avvento di SAX, si è reso possibile l'adattamento di molti algoritmi di *pattern matching/discovery* esistenti al *task* di *anomaly detection*. In questo capitolo verranno descritte le tecniche attualmente esistenti e come è stato possibile realizzare il programma oggetto di questa tesi.

3.1 Algoritmi esistenti

Successivamente al grande impatto che ha avuto SAX all'interno della comunità scientifica, molti sono gli algoritmi di *anomaly detection* nati basandosi su tale tecnica di discretizzazione simbolica. I concetti sui quali essi si basano posso essere suddivisi in due gruppi:

- Ricerca delle prime k sottosequenze insolite (***discord***). Tale operazione può essere effettuata, per esempio, calcolando la distanza tra ciascuna sottostringa di lunghezza predeterminata rispetto a quelle che compongono l'intero segnale oppure basando la ricerca sulla frequenza di quest'ultime. Con il supporto di una euristica corretta, è possibile individuare stringhe rare in tempi brevi e con risultati soddisfacenti.

- Confronto con un modello rappresentante il "comportamento corretto" della serie temporale. Tale modello può essere costruito manualmente o in automatico e, successivamente, utilizzato per il confronto diretto scansionando i dati una volta sola. Tale tecnica si presta molto per applicazioni di tipo *real time* in quanto, una volta effettuata la fase di *prepocessing* il *matching* richiede tempo lineare per quasi tutte le misure di distanza esistenti.

Di seguito vengono descritti due algoritmi rappresentativi per le categorie appena presentate.

3.1.1 HOT SAX: ricerca di sequenze insolite

HOT SAX [23] è l'algoritmo di *discord discovery* ideato da alcuni degli autori di SAX. Esso prevede l'analisi, mediante finestra scorrevole, delle sottosequenze di una serie temporale discretizzata nella ricerca di *pattern* molto diversi da tutti gli altri. Tale analisi si avvale della definizione di ***non-self match*** al fine di escludere corrispondenze fuorvianti (*trivial matches*) generalmente localizzate in un intorno prossimo alla finestra in esame, che porterebbero a rilevazioni errate delle anomalie:

Non-self match Si consideri una serie temporale $T = t_1, ..., t_m$ e siano $C = t_p, ..., t_{p+n-1}$ e $M = t_q, ..., t_{q+n-1}$ due sottosequenze corrispondenti di T lunghe n. M è detto *non-self match* per C se $|p - q| \geq n$ (in pratica, le due sequenze non devono sovrapporsi).

Alla luce di tale definizione un *discord* può essere visto come la sottosequenza di una serie temporale con distanza maggiore dal suo *non-self match* più vicino. Il seguente pseudocodice illustra l'algoritmo di tipo *brute force* capace di trovare il *discord* in tempo $O(m^2)$.

```
1  Function  [dist, loc ]= Brute_Force(T, n)
2   best_so_far_dist = 0
3   best_so_far_loc = NaN
4
5  For p = 1 to |T| - n  + 1                          // Begin Outer Loop
6    nearest_neighbor_dist = infinity
7    For q = 1 to |T| - n  + 1                        // Begin Inner Loop
8      IF | p - q | ≥ n                               // non-self match?
9        IF  Dist (tp,...,tp+n-1,  tq,...,tq+n-1)  < nearest_neighbor_dist
10         nearest_neighbor_dist = Dist (tp,...,tp+n-1,  tq,...,tq+n-1)
11       End
12     End                                            // End non-self match test
13   End                                              // End Inner Loop
14   IF nearest_neighbor_dist > best_so_far_dist
15      best_so_far_dist = nearest_neighbor_dist
16      best_so_far_loc  = p
17   End
18 End                                                // End Outer Loop
19 Return[ best_so_far_dist, best_so_far_loc ]
```

In [23] gli autori implementano un algoritmo che, avvalendosi di tecniche euristiche ed a strutture dati quali dizionari ed alberi, riesce ad ottenere lo stesso risultato riducendo la complessità computazionale a $O(m)$.

3.1.2 Assumption free anomaly detector

Questo algoritmo, discusso dagli autori in [24] ed implementato come *applet* Java all'indirizzo web `http://alumni.cs.ucr.edu/~wli/SSDBM05/`, permette la ricerca di anomalie attraverso il calcolo del "grado di anormalità" di ciascun punto della serie temporale basandosi sulla rappresentazione *bitmap* discussa in sezione 2.4.4. In pratica, vengono create due finestre concatenate che scorrono l'intero segnale:

Lag window rappresenta quanta informazione del passato si vuole mantenere nella ricerca di *pattern* anomali.

Lead window indica quanto guardare in avanti mentre si scansiona la serie temporale.

Ciascuna di esse viene convertita nella rappresentazione SAX, successivamente vengono contate le frequenze di tutte le sottostringhe di entrambe le parole ottenendo così le due *bitmap* che le rappresentano. La distanza tra le due finestre, calcolata mediante l'equazione 2.8, indica il grado di "grado di anormalità" del punto individuato dalla loro intersezione, più questo valore è elevato più ci si sta avvicinando ad uno dei *pattern* cercati. La seguente figura mostra uno *screenshot* del programma di *anomaly detection* in esame nel quale è possibile osservarne il funzionamento.

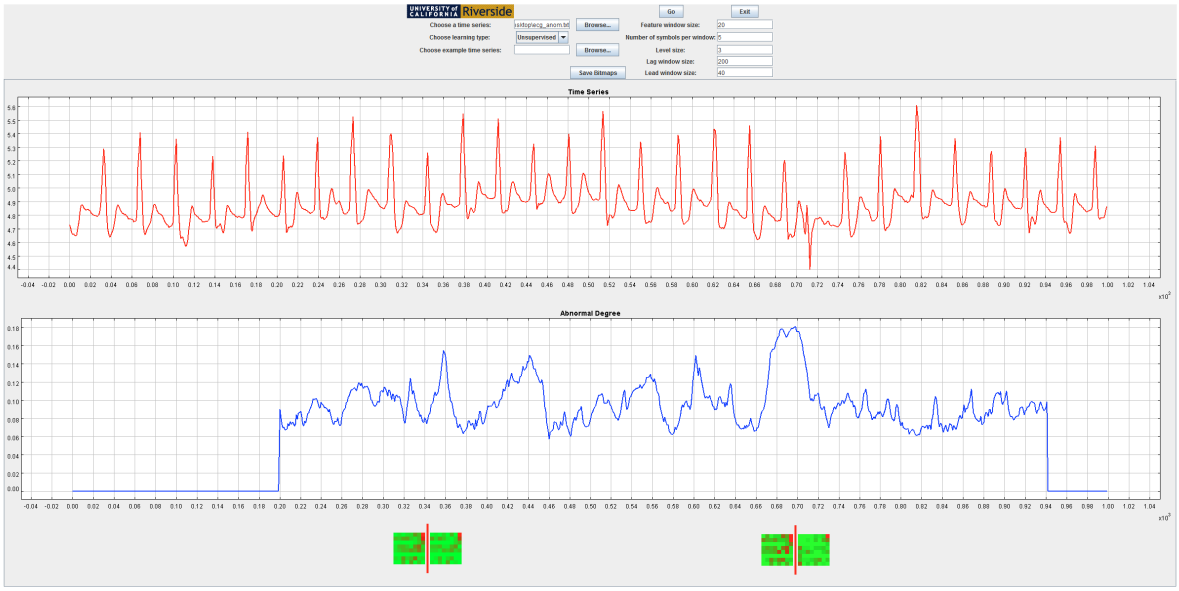

Figura 3.1: *Snapshot* del *tool* "assumption free anomaly detector". Nella banda inferiore è possibile osservare le *bitmap* corrispondenti alle finestre confrontate in un caso (**sinistra**) in cui esse sono simili e dunque il grado di anomalia è basso, ed in un altro (**destra**) in cui queste appaiono molto diverse.

Il programma prevede due modalità di funzionamento, che si differenziano nel modo in cui la *lag window* viene considerata:

Supervisionato: è possibile inserire in *input* due serie temporali. Una di queste verrà utilizzata interamente come *lag window* e dunque si richiede che in essa sia descritto il comportamento corretto del segnale da analizzare. La seconda serie è quella nella quale ricercare le anomalie.

Non supervisionato: è sufficiente impostare le dimensioni desiderate per le due finestre e quella del *pattern* cercato; l'algoritmo procede come descritto precedentemente.

È interessante osservare come la complessità computazionale dell'algoritmo sia lineare nel tempo. Ad ogni istante della scansione, infatti, è sufficiente introdurre un nuovo campione ed eliminare quello meno recente aggiornando così solo due *pixel* di ciascuna *bitmap*.

3.2 CADET: anomaly detection basato su tecniche di clustering

`CADET`, programma realizzato come lavoro di questa tesi, nasce dall'idea di unire la potenza dell'approccio SAX e delle tecniche di *clustering* al problema della ricerca di anomalie su serie temporali. Differentemente dagli algoritmi precedentemente discussi, esso ha come obiettivo

primario non quello di cercare direttamente l'anomalia, bensì la costruzione di un modello robusto per ciò che è considerabile come andamento corretto del segnale di ingresso. Solo successivamente all'individuazione di tale modello, che può essere costituito da uno o più *pattern* significativi, viene eseguita la vera e propria fase di *anomaly detection*. Così come per il *tool* presentato in sezione 3.1.2, `CADET` permette l'esecuzione in modalità supervisionata, qualora sia possibile etichettare in anticipo i vari *frame* componenti la serie di *input* come anomali o meno. Nel caso tale informazione non sia presente, il *software* crea autonomamente un *voting set* da utilizzare per la costruzione del modello. Una caratteristica innovativa, finora mai introdotta, consiste nella possibilità di esportare il modello generato con la conseguente capacità di utilizzare come *test set* una serie temporale differente da quella di ingresso[1]. Poiché il programma è stato pensato per un utilizzo su dati il cui asse dei tempi è su scala giornaliera (ipotesi che verrà successivamente generalizzata a *frame* di qualsiasi dimensione), esso non prevede l'utilizzo di una *sliding window* che avanza di un campione per volta, così come i precedenti studi sull'argomento, ma di una finestra che analizza i singoli giorni senza intersezioni. In questo modo si è potuto evitare di ricadere nel caso di *meaningless clustering* discusso in [25]. La seguente immagine mostra la *pipeline* completa eseguita dall'applicativo sviluppato; ciascuna fase verrà descritta in dettaglio nelle sezioni successive.

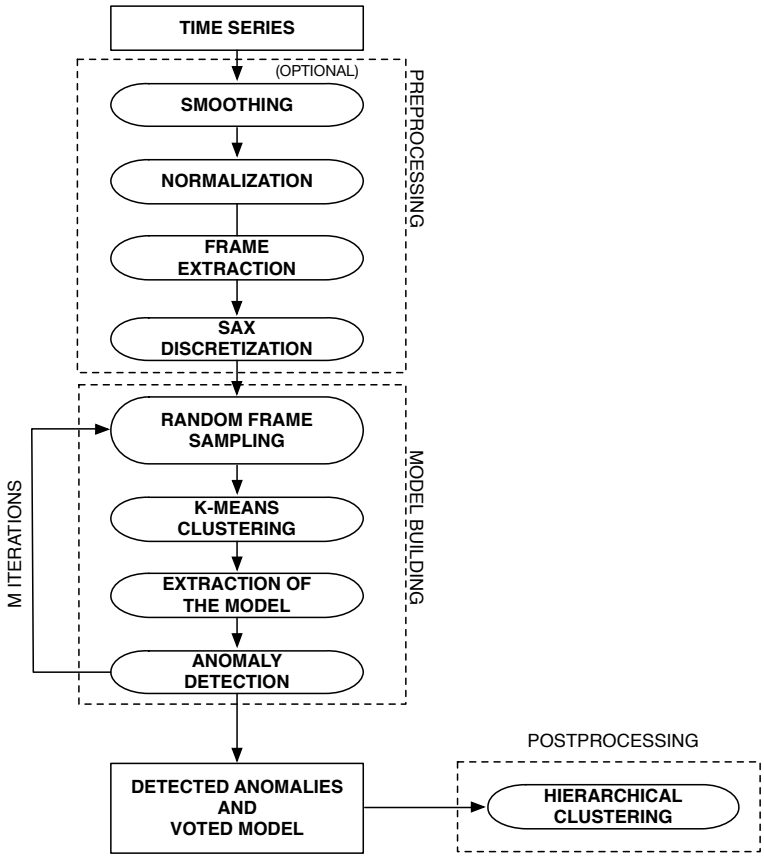

Figura 3.2: *Pipeline* esecutiva dell'intero processo di *anomaly detection*.

[1]Ovviamente, i dati da analizzare successivamente devono rispecchiare la stessa natura di quelli utilizzati per la costruzione del modello. Immaginando, per esempio, di applicare tale funzionalità su dati di temperature si può pensare a serie temporali ottenute da sensori differenti o in periodi di tempo diversi.

3.2.1 Preprocessing

I dati reali spesso soffrono di problematiche che ne rendono complicata l'analisi diretta: valori mancanti, *outlier* e rumore sono le cause principali che possono compromettere le prestazioni di un algoritmo di *data mining*. Si necessita, dunque, di una prima fase di *preprocessing* che prepari il segnale da analizzare rimuovendo eventuali disturbi e convertendolo nel formato desiderato. Nello studio delle serie temporali oggetto di questa tesi, si è reso utile prevedere un meccanismo di rimozione del rumore basato su filtraggio gaussiano; successivamente i singoli giorni vengono estratti per essere, infine, discretizzati secondo la rappresentazione SAX.

3.2.1.1 Rimozione del rumore tramite filtraggio gaussiano

Come mostrato nell'immagine 3.3, le serie temporali ottenute tramite sensori termici sono soggette ad errori di misura che tendono a dare una forma oscillata al segnale generato. Per limitare il contributo di queste variazioni continue presenti nelle serie temporali da analizzare, `CADET` applica un primo filtraggio mediante *smoothing* di tipo gaussiano. Questo può essere visto come una variante del *box filter*, che sostituisce ciascun punto di un segnale con il valore atteso dei campioni appartenenti ad un suo intorno (*smoothing window*) di dimensione specificata. Nel filtraggio gaussiano, viene assegnato un peso ai punti appartenenti alla finestra esaminata secondo una distribuzione normale: più essi sono lontani dal centro, minore è il contributo apportato alla media. Formalmente, l'*output* di tale operazione è ottenuto tramite convoluzione tra la serie di ingresso ed il segnale $G(t) = \frac{1}{\sqrt{2\pi\sigma^2}} \cdot e^{-\frac{(x-\mu)^2}{2\sigma^2}}$.

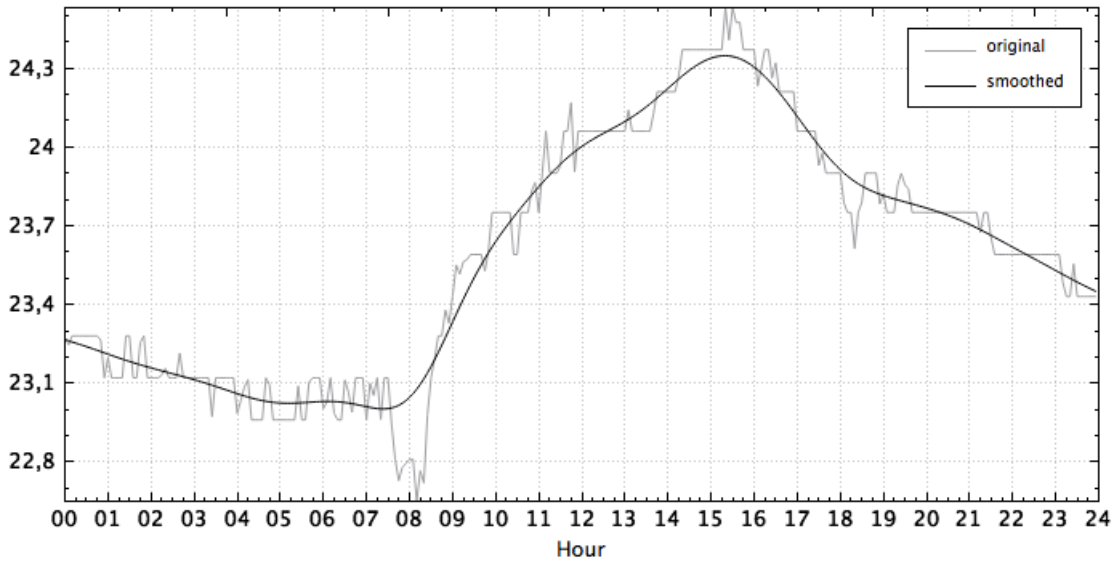

Figura 3.3: Serie temporale rappresentante la temperatura rilevata da un sensore termico nell'arco di una giornata. Si osservi come, successivamente al filtraggio gaussiano, sia più semplice comprendere l'andamento naturale dei dati.

Come confermato successivamente da prove sperimentali sui dati, questa fase può essere ritenuta facoltativa; l'algoritmo SAX, infatti, applica (implicitamente) un filtraggio analogo durante la trasformazione nella rappresentazione PAA dei dati.

3.2.1.2 Normalizzazione

Come già anticipato, l'idea sulla quale si basa `CADET` è quella di far avanzare una finestra scorrevole che analizzi giorno per giorno l'intera serie temporale fornita in ingresso alla ricerca di anomalie. Se si applicasse l'algoritmo SAX direttamente su ciascun *frame* estratto, operazione svolta da praticamente tutti i *tool* attualmente esistenti, il risultato sarebbe quello di analizzare la forma del segnale in esso contenuto e non i valori effettivi assunti in relazione all'intera serie di rilevazioni. Poiché può essere considerato utile mantenere l'informazione sul *trend* della serie temporale, in fase di sviluppo si è deciso di diversificare lo *step* di normalizzazione secondo le seguenti scelte progettuali:

Normalizzazione dell'intera serie temporale: l'intera serie viene normalizzata secondo l'equazione 2.1 e successivamente i singoli giorni vengono estratti e discretizzati mediante l'algoritmo SAX.

Normalizzazione per mesi: ciascun mese viene normalizzato singolarmente di modo da far mantenere il più possibile ai dati eventuali caratteristiche di tipo stagionale.

Normalizzazione per settimane: la normalizzazione viene applicata localmente su ciascuna settimana, in questo modo è possibile analizzare meglio i festivi che, generalmente, hanno andamenti diversi rispetto ai giorni feriali.

Normalizzazione per giorni: è la classica strategia nella quale l'algoritmo SAX (comprensivo di normalizzazione) viene applicato localmente su ciascuna sequenza estratta.

La seguente immagine mostra i risultati ottenibili tramite alcune delle strategie sopra descritte:

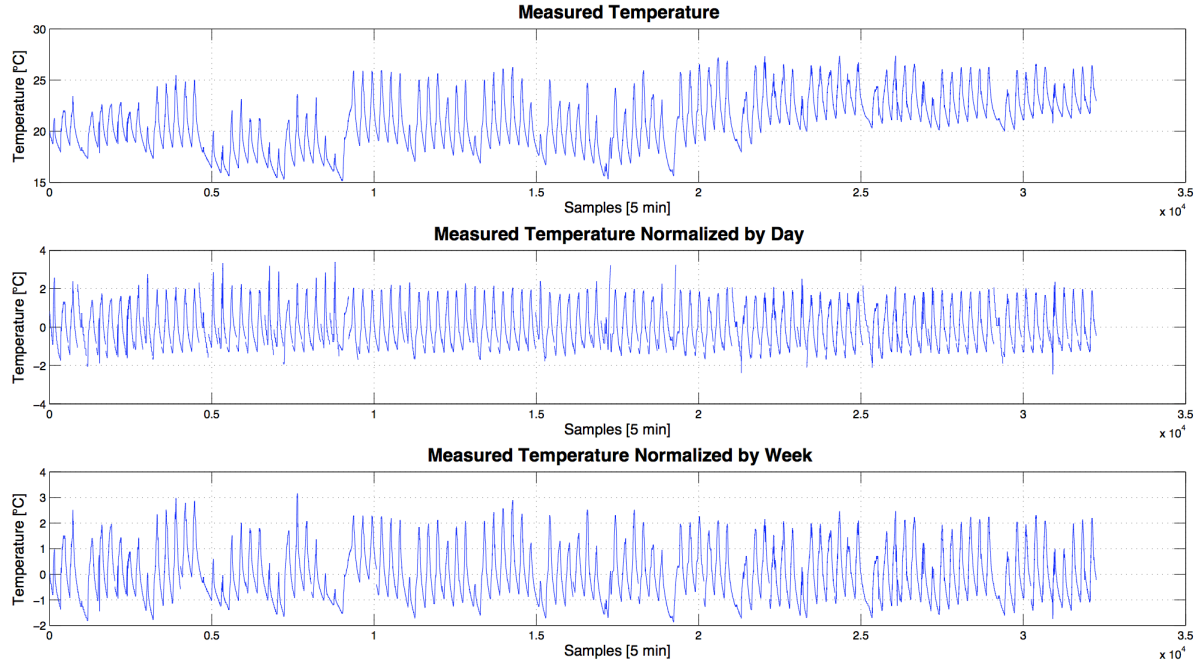

Figura 3.4: Risultato di alcune delle tecniche di normalizzazione descritte in questa sezione. Si osservi come, a seconda della procedura scelta durante la fase di *preprocessing*, l'informazione sull'andamento dei dati venga radicalmente modificata.

In figura 3.5 è possibile osservare l'importanza di una scelta di normalizzazione adeguata; essa, infatti, condizionerà il modo in cui i dati verranno interpretati nelle fasi successive. Se si analizzano le immagini sottostanti, rappresentanti l'andamento in due giorni diversi delle temperature rilevate all'interno di un ufficio di una scuola elementare, è possibile notare che i due segnali presentano una forma del tutto simile, nonostante siano costituiti da valori su intervalli diversi. Questo perché in entrambi i giorni il riscaldamento della stanza era in funzione, ma solo in uno dei due (l'immagine di sinistra) l'ufficio conteneva del personale. Normalizzando per giorno i due segnali genererebbero due parole SAX aventi una distanza molto piccola, cosa che non avverrebbe se, per esempio, si normalizzasse per mese. Spetta all'utilizzatore finale, a seconda dell'obiettivo dell'analisi da effettuare (rilevare i giorni in cui il riscaldamento era acceso inutilmente o semplicemente quelli in cui quest'ultimo era in funzione), la decisione su quale strategia di normalizzazione utilizzare.

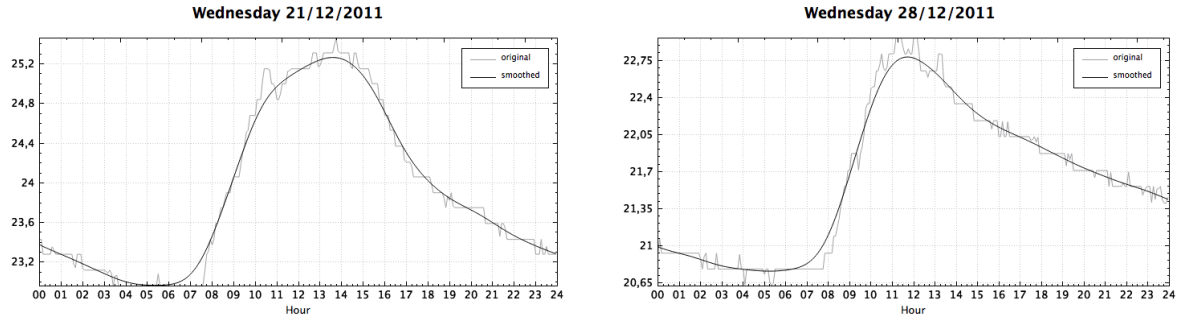

Figura 3.5: Esempio di due *pattern* aventi la stessa forma ma con valori diversi. A seconda della scelta presa in fase di normalizzazione, essi saranno interpretati come simili o come differenti una volta discretizzati e confrontati.

3.2.1.3 Estrazione dei giorni e discretizzazione SAX

Successivamente alla fase di normalizzazione della serie temporale, i dati sono pronti per essere suddivisi nelle finestre giornaliere che, una volta convertite nella notazione SAX, costituiranno il *dataset* di *input* della procedura di *anomaly detection* in esame. Poiché l'algoritmo di *clustering* che verrà descritto in seguito è stato progettato per operare su dati appartenenti all'insieme dei numeri reali, per la rappresentazione interna dei *pattern* da analizzare si è scelto di utilizzare l'alfabeto $\Sigma = \{\alpha_1, ..., \alpha_a\}$ con $\alpha_i \in \mathbb{N}$.

3.2.2 Costruzione del modello basata su consenso e ricerca di anomalie

Ora che i dati sono pronti per l'elaborazione è possibile passare alla fase di costruzione del modello rappresentante l'evoluzione corretta della serie temporale. L'intera lista di parole SAX rappresentante i giorni estratti dalla serie temporale viene inizialmente sfoltita per creare un *voting set* per l'algoritmo di *clustering* (di *default* esso conterrà il 70% dei giorni totali). Una volta suddiviso tale insieme in *cluster*, il centroide di ciascuno di essi rappresenta un *pattern* per il modello in costruzione. Grazie al confronto diretto con tale modello i giorni appartenenti al segnale di ingresso vengono, infine, etichettati come anomali o meno.
La procedura precedentemente descritta viene reiterata per un numero di volte specificabile dall'utente; ad ogni esecuzione verrà individuato un certo insieme di *pattern* anomali e solo quelli la cui frequenza di rilevazione supera un controllo mediante soglia predeterminata costituiranno l'*output* del processo di *anomaly detection*. Quest'ultima fase è realizzata mediante una procedura che permette di votare il modello che ha individuato il maggior numero di anomalie.

3.2.2.1 Creazione del voting set

Prima di procedere alla costruzione di un modello viene selezionato, con probabilità uniformemente distribuita, un sottoinsieme dei dati di *input*. Questa operazione permette di rendere l'algoritmo ripetibile e meno sensibile al rumore. Se si sta eseguendo `CADET` in maniera non supervisionata, eventuali anomalie presenti nel *voting set* verranno rilevate ed eliminate nelle fasi successive. La modalità supervisionata, invece, permette all'utente di specificare se nel *dataset* di ingresso siano presenti dei giorni anomali, di modo che essi vengano esclusi fin da subito per non interferire durante la costruzione del modello. È possibile, infine, fornire al *software* un calendario di date da trascurare, per esempio i giorni festivi, che possono essere ritenute fuorvianti per l'analisi.

3.2.2.2 k-means clustering

Una volta generato il *voting set* è possibile eseguire l'algoritmo **k-means**, necessario per la costruzione di un modello che rappresenti la serie temporale. Questo è un algoritmo di *clustering* partizionale che permette di suddividere un insieme di punti multidimensionali in k gruppi sulla base dei loro attributi. Ogni *cluster* C_i viene identificato mediante un centroide c_i, calcolato come punto medio degli elementi in esso contenuti. L'obiettivo che l'algoritmo si prepone è di minimizzare la distanza totale *intra-cluster*, ossia la somma delle distanze di ciascun punto con il centroide ad esso associato. La seguente equazione formalizza la definizione della funzione obiettivo utilizzata:

$$\min \sum_{i=1}^{k} \sum_{A \in C_i} \text{dist}(A, c_i) \tag{3.1}$$

L'algoritmo segue una procedura di ricerca locale iterativa:

1. crea k partizioni ed assegna a ciascuna di esse i punti d'ingresso casualmente o usando alcune informazioni euristiche.
2. calcola il centroide di ogni gruppo.
3. costruisce una nuova partizione associando ciascun punto di *input* al *cluster* il cui centroide è più vicino.
4. i punti 2 e 3 vengono reiterati finché l'algoritmo non converge (ad esempio dopo un certo numero di iterazioni).

In figura 3.6 viene mostrato un esempio di esecuzione della procedura sopra descritta.

`CADET` utilizza un implementazione modificata di un algoritmo in `C++` presente all'indirizzo internet `http://infolab.stanford.edu/~loc/localsearch.cxx` che, ispirandosi agli studi in [26], permette di determinare il miglior valore di k: l'algoritmo viene eseguito più volte, per diversi valori del numero di *cluster*, e la soluzione finale sarà quella con il valore minore della funzione obiettivo.

In via sperimentale, per la definizione di funzione di distanza da utilizzare per la funzione obiettivo (3.1) sono state effettuate due prove:

- creazione delle *bitmap* per ciascun punto ed utilizzo della (2.8) come misura di similarità
- utilizzo diretto della distanza definita da SAX (2.6)

La scelta finale è ricaduta sul secondo approccio in quanto più efficiente e con risultati migliori. L'utilizzo di *bitmap* infatti, oltre a fornire risultati scarsi e per niente rappresentativi, comporta l'aggiunta di un parametro (la dimensione dell'icona) che ne condizionava la complessità computazionale. Una delle cause principali per le basse prestazioni ottenibili è da attribuire anche alle limitazioni imposte sulla taglia dell'alfabeto, che deve essere costituito da un numero di elementi che sia una potenza di due.

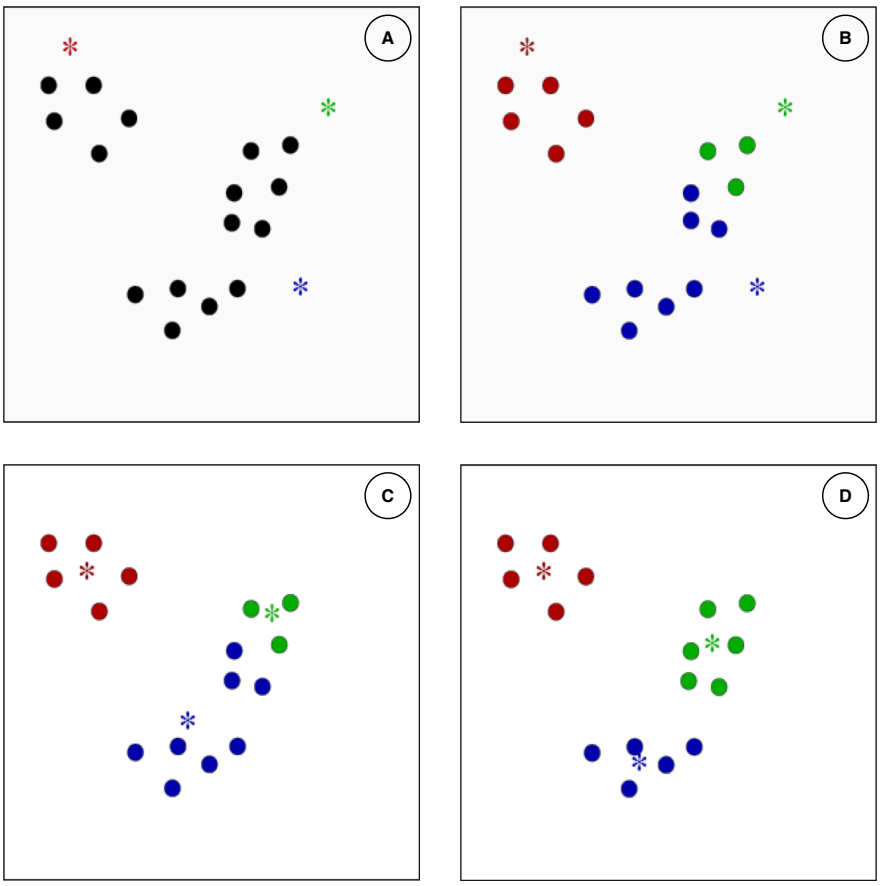

Figura 3.6: Esempio di esecuzione dell'algoritmo *k-means clustering* su un insieme di punti bidimensionali con $k = 3$. Alla prima iterazione i tre centroidi vengono generati casualmente (**A**) ed assegnati ai punti corrispondenti (**B**), successivamente (**C**) i *cluster* vengono ricalcolati e la procedura viene reiterati finché l'algoritmo non converge (**D**).

3.2.2.3 Costruzione del modello e rilevazione delle anomalie

L'*output* fornito dall'algoritmo k-means è costituito dall'insieme dei centroidi di tutti i *cluster* da esso individuati. Quello che `CADET` vuole costruire a questo punto è un modello robusto e ben rappresentante l'andamento corretto della serie temporale in analisi; per questo, i *cluster* con una cardinalità troppo bassa (rispetto ad una soglia definibile a priori) vengono scartati. Infine, poiché nel *voting set* potrebbero esserci stati presenti anomalie sistematiche, vengono rimossi anche i centroidi che appaiono troppo distanti da tutti gli altri.
Una volta ottenuto il modello cercato, esso può essere esportato per utilizzi successivi o riapplicato all'intera serie temporale fornita in *input*. In entrambi i casi la ricerca di anomalie viene eseguita calcolando la distanza minima tra ciascun giorno discretizzato e tutte le parole SAX

contenute nel modello finale. Se tale distanza è superiore ad una soglia fissata per alcuni giorni, essi verranno etichettati come anomali e forniti come *output* del processo di *anomaly detection*. Formalmente, dato l'insieme $W = \{w_1, ..., w_n\}$ di stringhe, il modello $P = \{p_1, ..., p_m\}$ ed una soglia Z, il giorno rappresentato da w_i viene considerato anomalo se:

$$\min_{1 \leq j \leq m} \text{SAX Dist}(w_i, p_j) > Z \tag{3.2}$$

3.2.3 Reiterazione e sistema di voto

Operando in modalità non supervisionata, bisogna essere consapevoli che è molto probabile che il *voting set* contenga delle anomalie. Queste possono essere erroneamente assegnate a *cluster* contenenti "giorni validi", o essere raggruppate assieme qualora la loro rappresentazione sia molto simile (ad esempio le domeniche nel *dataset* obiettivo di questa tesi). Questa situazione è inevitabile poiché nelle applicazioni pratiche non si ha nessuna conoscenza a priori sui dati e quindi non è possibile distinguere tra centroidi che rappresentano comportamenti desiderati e quelli che individuano le anomalie. Per aggirare tale problematica, l'algoritmo di *clustering* viene reiterato M volte su diversi *voting set*, ciascuna delle quali costruirà un modello M_i necessario all'individuazione delle corrispondenti anomalie L_i. Poiché l'output di `CADET` è costituito da un unico modello (contenente uno o più *pattern*) viene eseguita una procedura di voto necessaria a eleggere il migliore tra quelli generati in ciascuna iterazione. Tale processo è costituito dai seguenti passaggi:

1. Si costruisce una lista di anomalie che sono state rilevate per almeno un numero $q \leq M$ di volte.
2. Ciascuna anomalia individuata al punto precedente vota i modelli che l'avrebbero individuata in funzione della soglia di distanza Z definita in 3.2.
3. il modello che ha ottenuto il maggior numero di voti viene eletto come modello finale.
4. in caso di pareggio tra modelli diversi viene scelto quello la cui somma delle distanze tra i *pattern* in esso contenuti e anomalie da questi individuate è massima.

3.2.4 Postprocessing

Un'interessante funzionalità aggiuntiva di `CADET` permette la clusterizzazione delle anomalie individuate durante la fase di *data mining* al fine di comprenderne meglio la distribuzione reciproca nello spazio vettoriale creato per l'algoritmo *k-means*. Per effettuare tale analisi il *tool* utilizza un algoritmo di ***clustering* gerarchico**. Esso genera una struttura ad albero (dendogramma) che fornisce una rappresentazione grafica del processo di raggruppamento delle istanze di *input*, rappresentata in figura 3.7 per una particolare esecuzione su dati di interesse per questa tesi.

Le strategie per il *clustering* gerarchico sono tipicamente di due tipi:

Agglomerativo: si tratta di un approccio *bottom up* in cui si parte dall'inserimento di ciascun elemento in un gruppo differente e si procede, successivamente, all'accorpamento graduale di *cluster* a due a due.

Divisivo: si tratta di un approccio *top down* in cui tutti gli elementi si trovano inizialmente in un singolo *cluster*, che viene via via suddiviso ricorsivamente.

Per questa fase di *postprocessing*, `CADET` si avvale di un implementazione *open source* in `Java` modificata dell'algoritmo agglomerativo presente all'indirizzo `https://github.com/lbehnke/hierarchical-clustering-java`. Essa richiede in *input* una matrice di distanze tra coppie di elementi per decidere quali *cluster* combinare, ed un criterio di collegamento che specifica la dissimilarità di due gruppi di punti come funzione della distanza a coppie tra elementi nei due insiemi. La matrice di distanza d viene calcolata mediante la onnipresente (2.6) mentre per i criteri di collegamento, dati due *cluster* A e B, è possibile scegliere tra le seguenti tre varianti per il calcolo della loro distanza $D(A, B)$:

Average linkage: $D(A,B) = \frac{1}{|A|\cdot|B|} \sum_{a\in A} \sum_{b\in B} d(a,b)$

Single linkage: $D(A,B) = \min_{a\in A,\, b\in B} d(a,b)$

Complete linkage: $D(A,B) = \max_{a\in A,\, b\in B} d(a,b)$

L'analisi manuale del grafico ottenuto può essere considerata l'operazione finale del processo di *anomaly detection*.

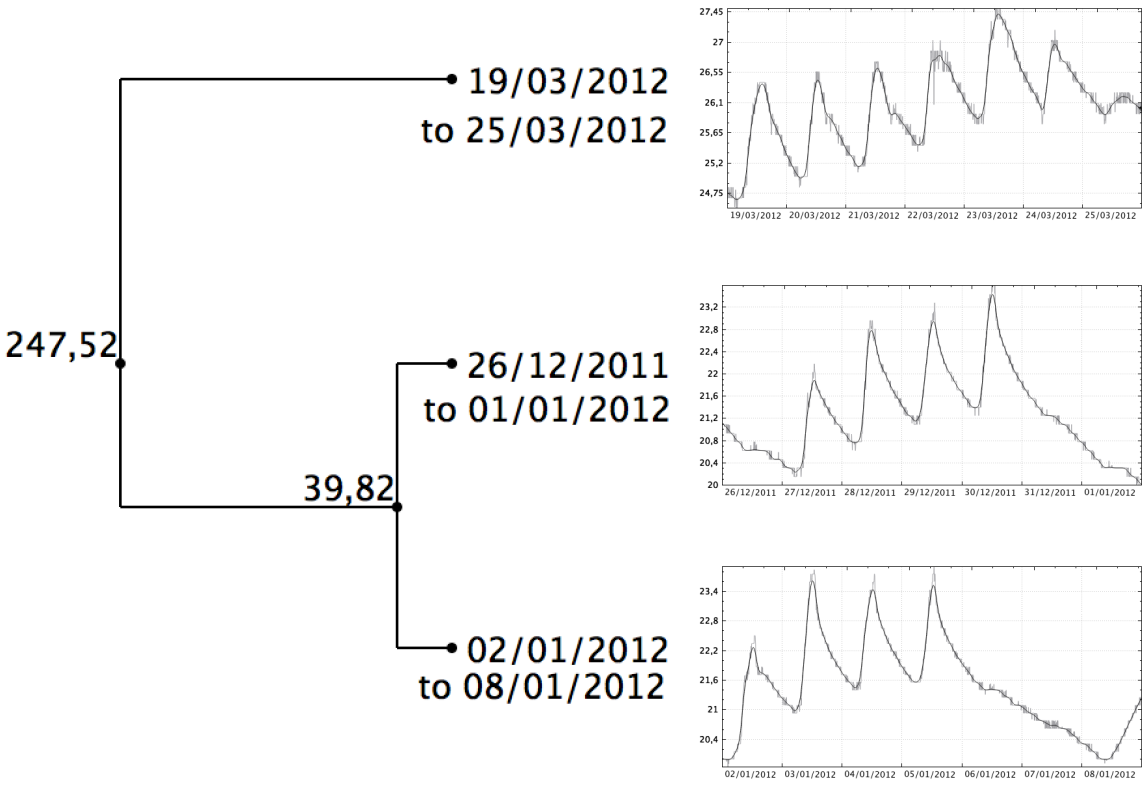

Figura 3.7: Esempio di esecuzione dell'algoritmo di *clustering* gerarchico su un insieme di anomalie settimanali rilevate da `CADET`. I numeri sul dendogramma rappresentano le distanze tra *cluster* corrispondenti.

Capitolo 4

Risultati sperimentali

In questo capitolo verranno presentati i risultati degli esperimenti eseguiti al fine di validare la procedura utilizzata da `CADET` per individuare le anomalie presenti in una serie temporale. Si vedrà, inoltre, come sia possibile generalizzare tale approccio per analizzare *pattern* di dimensioni diverse da quella giornaliera. Il metodo, inizialmente applicato sui dati forniti come obiettivo di questo lavoro di tesi, verrà testato anche su *dataset* differenti in maniera da facilitarne il confronto con i *tool* illustrati nel paragrafo 3.1.

4.1 Analisi dei dati di temperature della scuola Rigotti

I dati sui quali si è focalizzata l'implementazione di `CADET`[1], largamente discussi nel capitolo introduttivo, sono stati oggetto di numerosi esperimenti. Molti di essi si sono resi utili per comprendere al meglio quali fossero i parametri migliori da utilizzare nella fase di discretizzazione. Nelle sezioni seguenti verranno presentati i test realizzati per la ricerca di anomalie giornaliere e settimanali; verrà inoltre mostrato come l'utilizzo della modalità supervisionata permetta di analizzare dati di sensori estremamente rumorosi attraverso la costruzione di un modello solido basato su segnali il cui andamento rispecchia meglio il comportamento corretto che dovrebbe avere la serie temporale. L'intera analisi sarà incentrata, oltre che sull'analisi dei risultati ottenuti, sulla fase di *parameter tuning* della taglia dell'alfabeto, della dimensione dei dati discretizzati e sulla scelta della tecnica di normalizzazione.

4.1.1 Analisi giornaliera

Durante l'analisi di *pattern* di dimensione giornaliera si è osservato come la variazione del valore della **taglia per l'alfabeto** Σ risulti piuttosto ininfluente se questo viene scelto all'interno del *range* $(5 - 16)$. Questo fatto può essere giustificato dalla forma piuttosto simile posseduta dall'andamento di tutti i giorni non anomali estraibili dalle serie temporali in esame; poiché a valori più grandi corrisponde una sensibilità maggiore nella rilevazione di anomalie spetta all'utilizzatore la decisione su tale parametro. Nel caso di questa tesi un alfabeto di 6 simboli si è rivelato la scelta migliore su tutti i sensori analizzati. Per quanto riguarda il **numero di simboli** coi quali rappresentare i giorni, si è scelto di utilizzare una dimensione pari a 24, corrispondente all'idea di assegnare un carattere ad ogni ora della giornata. Con valori inferiori si corre il rischio di far risultare simili stringhe che in realtà sono molto distanti mentre, utilizzando valori superiori ci si potrebbe imbattere nel problema inverso. Il **metodo di normalizzazione** scelto, infine, è quello giornaliero. Si è notato come una analisi che confronti solo gli andamenti dei

[1] Dati di sensori termici le cui temperature sono rilevate ogni 5 minuti. La serie temporale copre un periodo di circa 4 mesi (08/12/2011 - 28/03/2012).

dati risulti più efficiente di una che preveda una correlazione con i valori che la serie temporale assume. Uno studio approfondito ha dimostrato come, applicando tale tecnica ai dati in esame, il risultato sia quello di individuare come anomali tutti quei giorni in cui il riscaldamento nell'aula analizzata sia spento: prevedendo l'utilizzo di un semplice calendario, dunque, è possibile capire dove questa condizione sia desiderata o meno e rilevare le anomalie di conseguenza. Di seguito vengono mostrati alcuni *screenshot* che mettono a confronto le differenze tra i risultati ottenibili con le modalità supervisionata e non supervisionata.

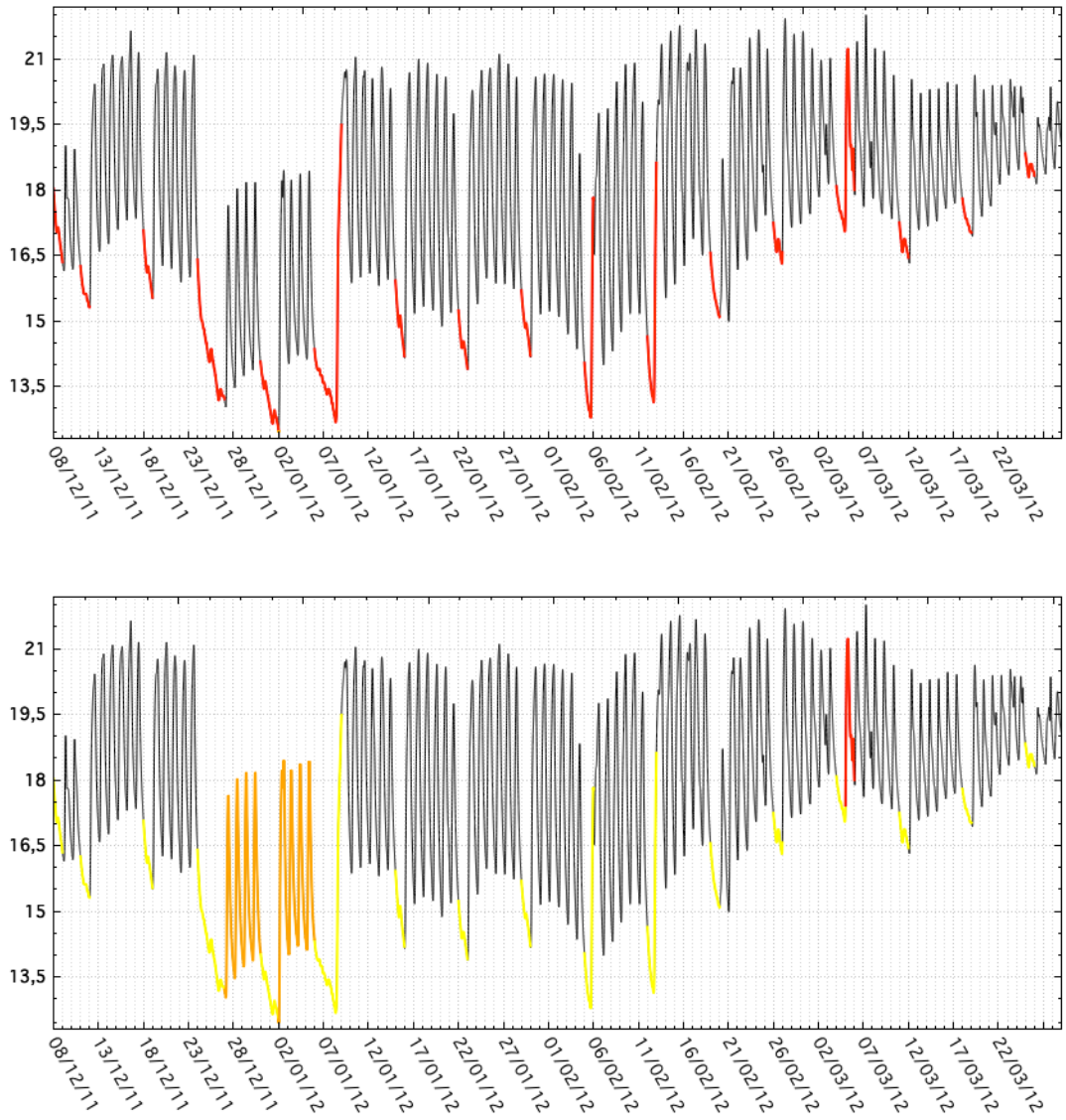

Figura 4.1: Analisi giornaliera della serie temporale riguardante le temperature rilevate da uno dei sensori presente all'interno dell'auditorium. Nella prima immagine, rappresentante l'esecuzione in modalità non supervisionata, le anomalie corrispondono ai giorni evidenziati in rosso. Nella seconda, rappresentante l'esecuzione in modalità supervisionata, i giorni in giallo (*warning*) corrispondono alle anomalie correttamente rilevate mentre quelli in arancione rappresentano giorni in cui il riscaldamento è acceso in un giorno in cui la scuola dovrebbe essere chiusa (e quindi altrettanto anomali).

Confrontando le due immagini è possibile osservare come l'utilizzo di un calendario riesca a colmare l'incapacità della normalizzazione giornaliera nell'individuazione di giorni anomali in funzione dei valori effettivi assunti da questi. Sicuramente una normalizzazione più ampia permetterebbe di rilevare tali anomalie ma essa sarebbe estremamente dipendente dalla lunghezza della serie temporale e dal periodo che essa copre nell'arco dell'anno. Un osservazione può essere fatta anche in merito della stagionalità dei dati: poiché è molto probabile che il *trend* del segnale sia diverso tra i mesi invernali e quelli estivi un approccio che si basi sui valori effettivi assunti dalla serie può essere facilmente ingannato e fornire risultati scadenti e del tutto fuorvianti.

4.1.2 Analisi settimanale

Il passo successivo dell'analisi è stato quello di verificare se il *tool* sviluppato fosse in grado di generalizzare la dimensione dell'anomalia ricercata in modo da essere applicato ad analisi su *pattern* di lunghezza differente. In questa sezione verranno presentati i risultati ottenuti effettuando la ricerca di anomalie settimanali. La prima osservazione può essere rivolta alla forma della serie temporale: diversamente da quello che avviene a livello giornaliero, l'andamento di settimane differenti è difficilmente confrontabile in maniera diretta poiché queste possono risultare distanti anche se simili tra loro per via della diversa scala di valori assunti tra i giorni in queste contenuti. Questa problematica, che in parte giustifica la mancata scelta di normalizzare per settimana durante l'analisi giornaliera, è facilmente aggirabile adottando come soglia di distanza un valore che permetta di discriminare il numero di picchi presenti in ciascuna sottosequenza estratta. Per questo tipo di analisi un approccio di tipo manuale, ossia tentando di ottenere dei parametri funzionanti per una serie della quale si conoscono in anticipo le anomalie e successivamente utilizzando tali valori per segnali sconosciuti, ha garantito i risultati migliori. La figura 4.2 mostra uno *screenshot* dell'analisi settimanale effettuata sui dati di un sensore termico presente in un ufficio della scuola elementare Rigotti.

Dall'immagine precedente, e dal dendogramma ricavato dalle anomalie rilevate (figura 4.3) è possibile trarre le seguenti osservazioni:

- le due settimane che coprono il periodo dal 26/11/2011 al 08/01/2012 sono sicuramente anomale. I picchi in esse contenute sono quattro contrariamente ai sei presenti in tutte le settimane con comportamento *standard*. In questi due *pattern* anomali, inoltre, i valori delle temperature sono più bassi del normale; la causa è molto probabilmente la mancanza di personale all'interno dell'ufficio monitorato nel quale, però, il riscaldamento è rimasto inutilmente attivo.

- la settimana da 19/12/2011 al 25/12/2011 è stata correttamente rilevata come anomala vista la differenza di un giorno intero (l'ultimo picco mancante) col resto della serie temporale. Poiché, però, tale picco mancante corrisponde al giorno della vigilia di natale, si può interpretare tale comportamento del segnale come non anomalo. Questo tipo di analisi può essere effettuata solo dopo il processo eseguito da `CADET` in quanto è soggettiva e dipende dalle variazioni comportamentali che sono conseguenza di un calendario didattico prestabilito. L'utilizzo di quest'ultimo può sicuramente agevolare il *tool* rendendolo più "consapevole" di tali situazioni, previste in anticipo.

- nell'ultima settimana anomala (dal 30/01/2012 al 05/02/2012) i picchi sono esattamente sei, ma quello relativo al sabato è molto più basso del normale. Probabilmente in tale giorno il riscaldamento era stato lasciato acceso nonostante non vi fosse personale operativo in tale ufficio. La settimana, dunque, è da considerarsi anomala.

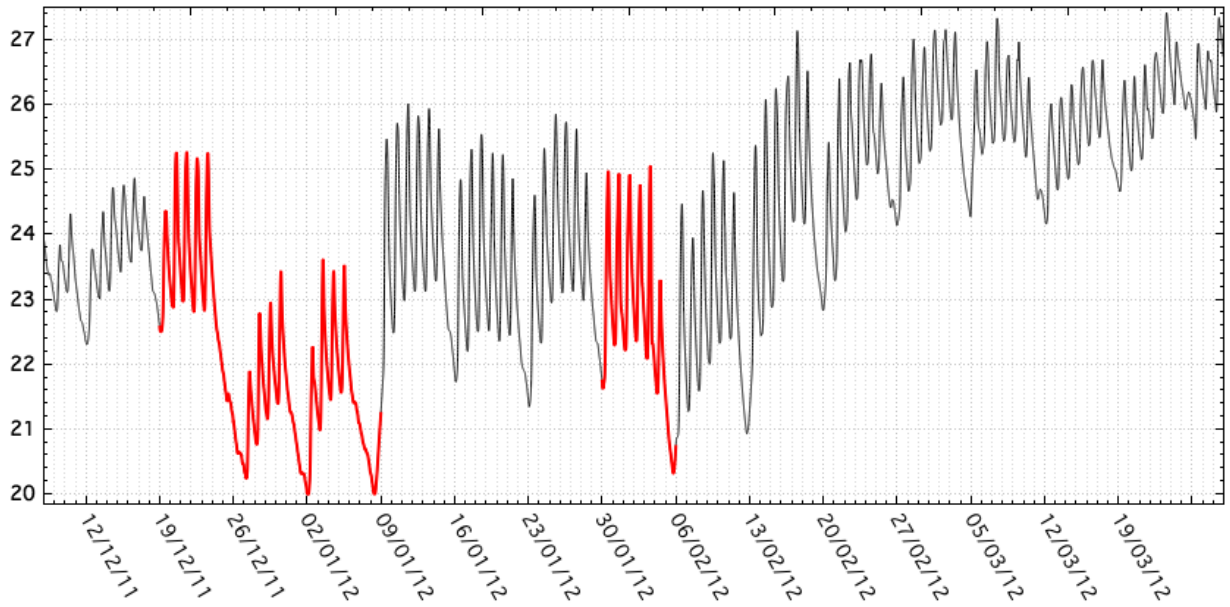

Figura 4.2: Analisi della serie temporale riguardante le temperature rilevate da uno dei sensori presente all'interno di un ufficio con dimensione del *pattern* pari ad una settimana.

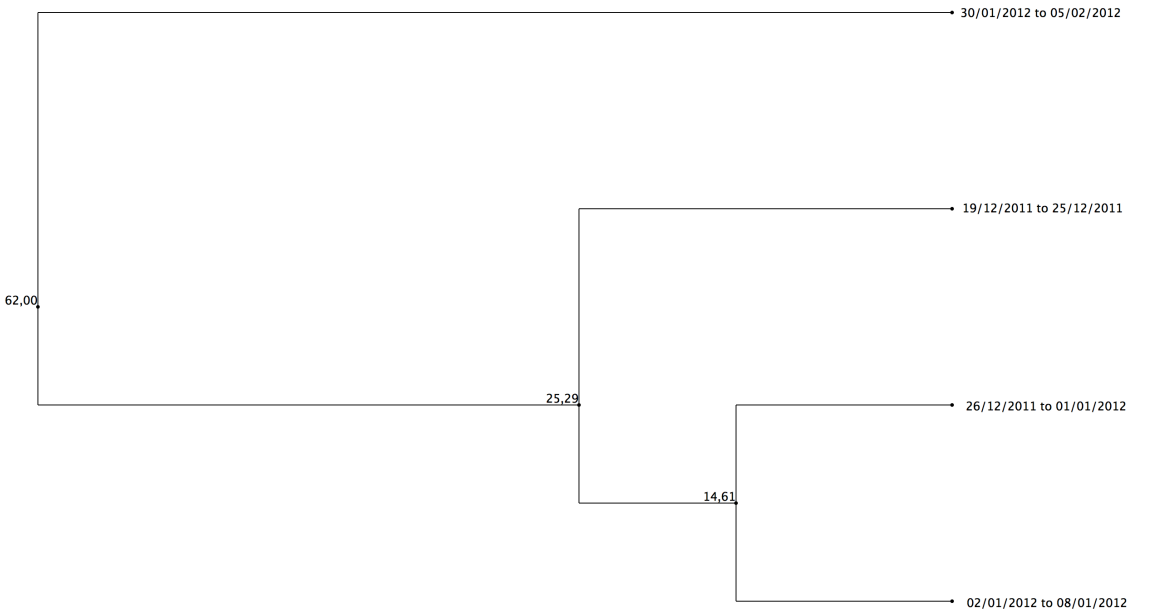

Figura 4.3: Risultato del *postprocessing* eseguito sulle anomalie rilevate dall'esecuzione di `CADET` sulla serie temporale di figura 4.2.

La seguente tabella riassume i parametri finali utilizzati per la ricerca di anomalie giornaliere e settimanali nei dati relativi alle temperature rilevate dai sensori presenti all'interno della scuola Rigotti:

	***Pattern* giornalieri**	***Pattern* settimanali**
Lunghezza stringhe SAX	24	168
Taglia dell'alfabeto	6	8
Normalizzazione	per giorno	tutta la serie
Soglia sulla distanza	9.6	40.0

Tabella 4.1: Parametri definitivi utilizzati da `CADET` per la ricerca di anomalie sulle serie temporali ricavate dai sensori della scuola elementare Rigotti. Il parametro relativo alla soglia sulla distanza è stato ricavato mediante studi sperimentali sui dati da analizzare.

Il valore di *default* per la soglia sulla distanza, modificabile dall'utente finale, è stato scelto come risultato di una funzione convessa che combina linearmente la lunghezza del *pattern* e la taglia dell'alfabeto secondo la seguente formula:

$$threshold = 0.2 \cdot pattern\ length + 0.8 \cdot alphabet\ size \tag{4.1}$$

Intuitivamente, tale scelta consente di tenere conto sia della discretizzazione sia della sensibilità della descrizione dei dati. Nel caso di analisi giornaliera, questo ha permesso di discriminare i *pattern* secondo la loro forma, mentre per la ricerca di anomalie settimanali tale parametro riesce a distinguere due sottosequenze in funzione dei picchi in esse contenuti.

4.1.3 Esportazione del modello

Una caratteristica aggiuntiva del *tool* realizzato permette di esportare il modello creato durante il processo di *anomaly detection*, il quale può essere riutilizzato in elaborazioni successive per evitare la fase di *clustering*. Qualora il modello costruito sia particolarmente robusto, la sua esportazione porta a benefici superiori a quelli ottenibili tramite l'utilizzo dell'applicativo in modalità supervisionata. Le applicazioni di tale funzionalità sono molteplici:

1. Costruzione di un modello robusto sui dati di un sensore specifico e applicazione di tale modello su serie temporali da esso generate, diverse da quella utilizzata per la fase di allenamento.

2. Costruzione di un modello robusto sui dati di un sensore e applicazione di tale modello su serie temporali generate da sensori differenti. Ovviamente, più i sensori sono simili migliori sono i risultati ottenibili: utilizzi consigliati prevedono, per esempio, sensori installati tutti all'interno della stessa aula o in stanze adibite allo stesso scopo (uffici).

3. Costruzione di un modello robusto sui dati di più sensori specifici, anche relativi a stanze differenti, con l'obiettivo di monitorare una serie temporale molto rumorosa il cui *training* risulterebbe complicato e con risultati non soddisfacenti.

Di seguito vengono riportati alcuni esempi di esecuzione per alcune delle tre applicazioni presentate.

Applicazione 2a - sensori installati nella stessa stanza

Per tale esperimento è stata presa in considerazione la stanza adibita ad auditorium. Come osservabile in figura 1.1 in essa sono stati installati ben sei sensori termici $(\Phi_1, ..., \Phi_6)$ ai quali

corrispondono i segnali $T_1, ..., T_6$. Poiché, in genere, tali serie sono fortemente correlate è possibile utilizzare una delle serie temporali T_i (e la sua rappresentazione SAX X_i) per costruire un modello M_i, il quale una volta esportato può essere utilizzato per analizzare una serie T_j differente. La figura 4.4 mostra i risultati ottenuti dall'esecuzione su entrambi i segnali analizzati. È interessante notare come le anomalie rilevate siano le stesse in entrambe le modalità; qualora esso sia ben rappresentante il comportamento della serie temporale, il vantaggio dell'utilizzo di un modello precalcolato è legato essenzialmente ai tempi di esecuzione, sicuramente inferiori poiché la fase di clusterizzazione è necessaria una sola volta a priori.

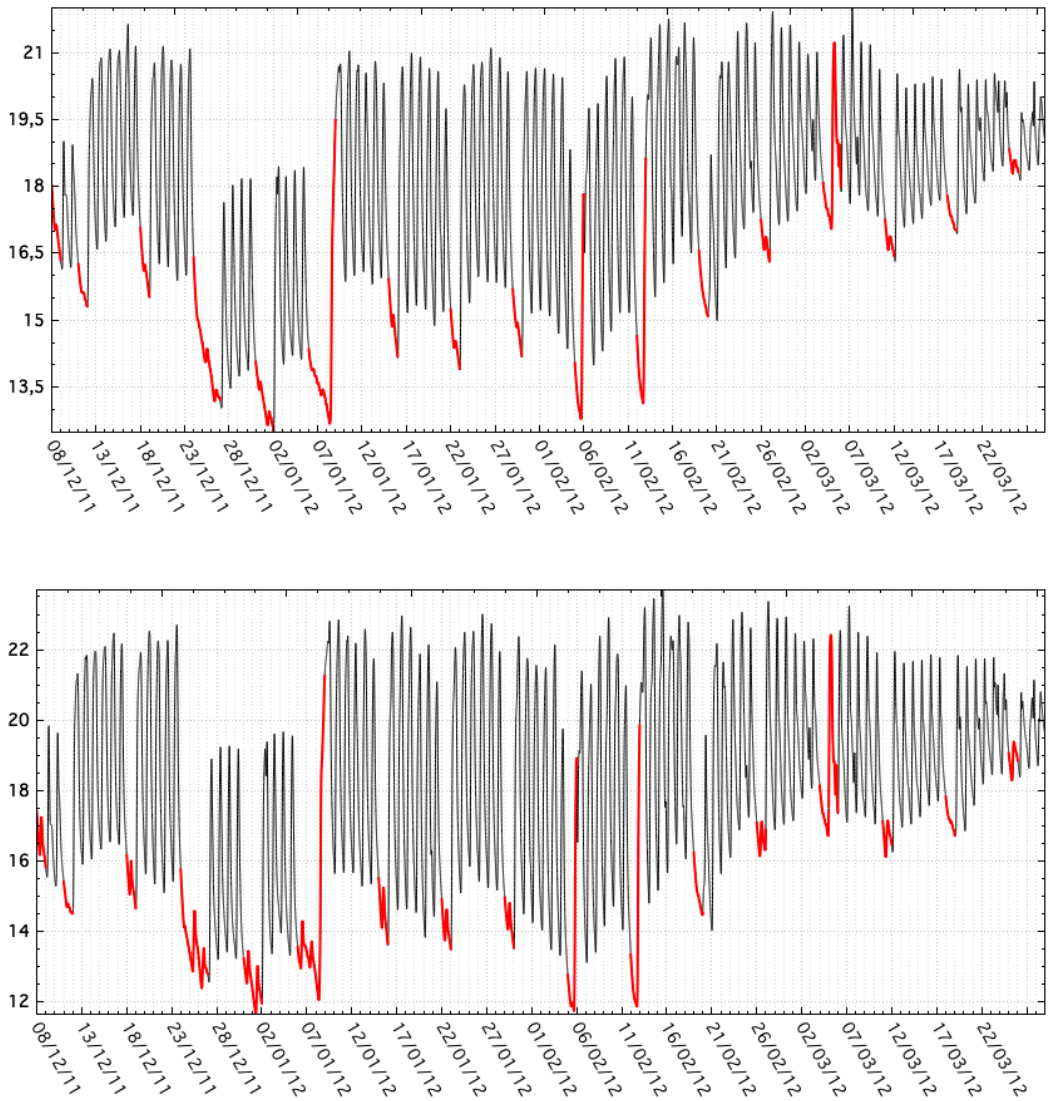

Figura 4.4: La prima immagine mostra il risultato dell'elaborazione in modalità non supervisionata sui dati di un sensore termico situato all'interno dell'auditorium della scuola elementare Rigotti. Il modello ricavato da tale esecuzione (`bbbbcdeeffffeeeeddcccbb`) è stato successivamente applicato ad un altro sensore installato nella medesima stanza (immagine sottostante). Se su quest'ultimo sensore fosse stata eseguita un analisi non supervisionata si sarebbe ottenuto il modello `aaabbceeffefeeeddccbbbba` con la conseguente individuazione delle medesime anomalie.

Applicazione 2b - sensori installati in aule differenti

Similarmente all'applicazione precedente, è possibile ottenere un modello M_i sui dati di un sensore Φ_i situato in una stanza A e, successivamente, applicare questo al segnale T_J ottenuto da Φ_j posto in un'aula differente B. Ovviamente le caratteristiche dei segnali devono essere simili e confrontabili. Per tale *test* sono stati analizzati i sensori presenti all'interno degli uffici della scuola elementare. Si è cercato di scegliere due aule distanti tra loro di modo da elaborare serie temporali il più differenti possibile. La figura 4.5 mostra i risultati ottenuti dall'esecuzione su entrambi i segnali analizzati. È interessante notare anche qui come le anomalie rilevate siano relative agli stessi giorni, seppure l'andamento della temperatura non fosse sempre analogo tra due uffici diversi.

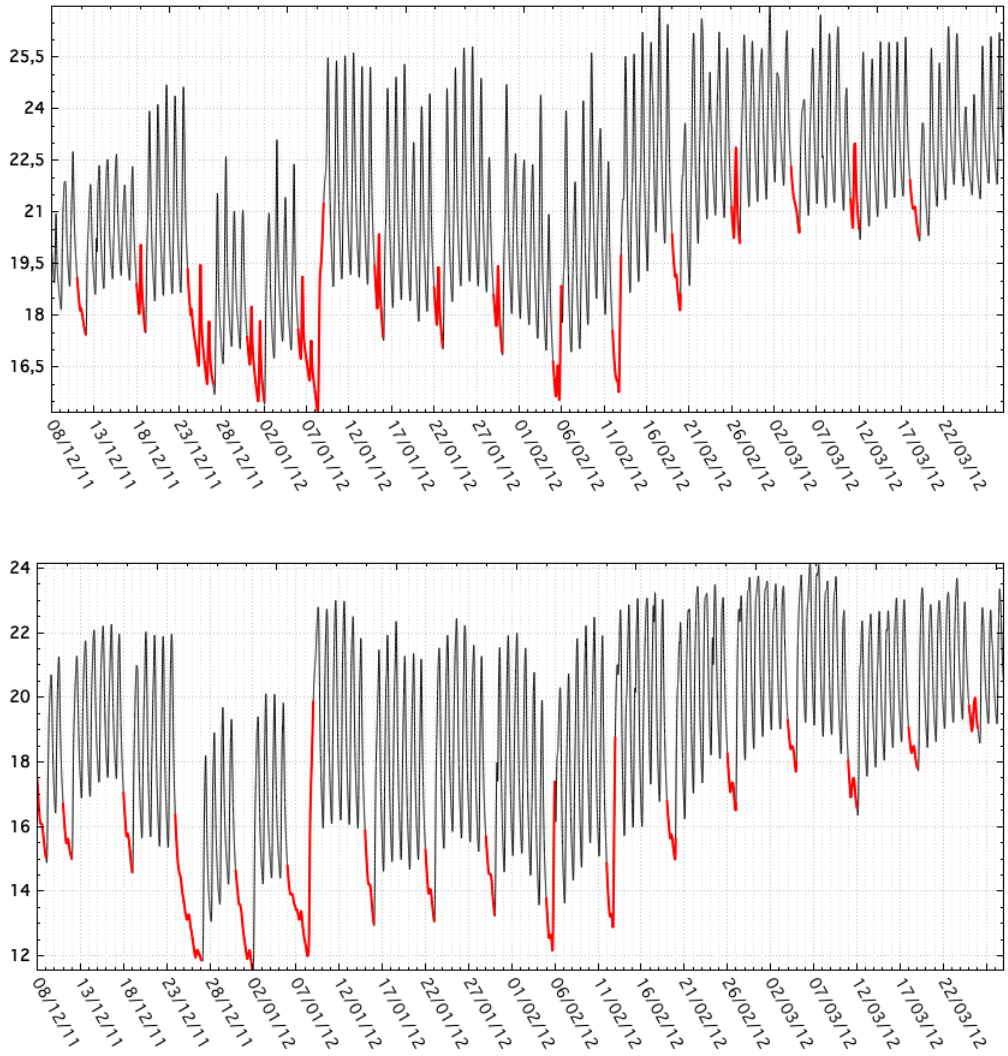

Figura 4.5: La prima immagine mostra il risultato dell'elaborazione in modalità non supervisionata sui dati di un sensore termico situato all'interno di un ufficio della scuola elementare Rigotti. Il modello ricavato da tale esecuzione (`bbbbbccdeeffffeeddcccbb`) è stato successivamente applicato ad un altro sensore installato su un ufficio differente (immagine sottostante). Anche in questo caso, se su quest'ultimo sensore fosse stata eseguita un analisi non supervisionata si sarebbe ottenuto il modello `aaaabbcdeeeffffeddccbbb` con la conseguente individuazione delle medesime anomalie.

Applicazione 3 - modello unificato

Il più interessante di tutti gli esperimenti è stato quello di creare un modello *multi-pattern* robusto per alcuni dei sensori sparsi il tutto il perimetro interno della scuola elementare Rigotti con l'obiettivo di riunire i *pattern* generati da ciascuno di essi in un unico modello capace di elaborare dati di sensori termici posti in aree affollate, come per esempio l'atrio. Quest'area della scuola, infatti, fornisce una serie temporale molto rumorosa la cui analisi diretta porterebbe solo alla rilevazione delle anomalie più semplici (figura 4.6b). A tale scopo sono stati utilizzati i modelli ricavati dai precedenti studi (uffici e auditorium) al quale è stato aggiunto quello ottenuto mediante la scansione del segnale relativo alla mensa (`bbbbcdeeeffffeeddccccbb`). In figura seguente è possibile apprezzare il confronto tra i risultati ottenuti, e quelli che si sarebbero avuti analizzando la serie temporale di un sensore installato nell'atrio in modalità non supervisionata.

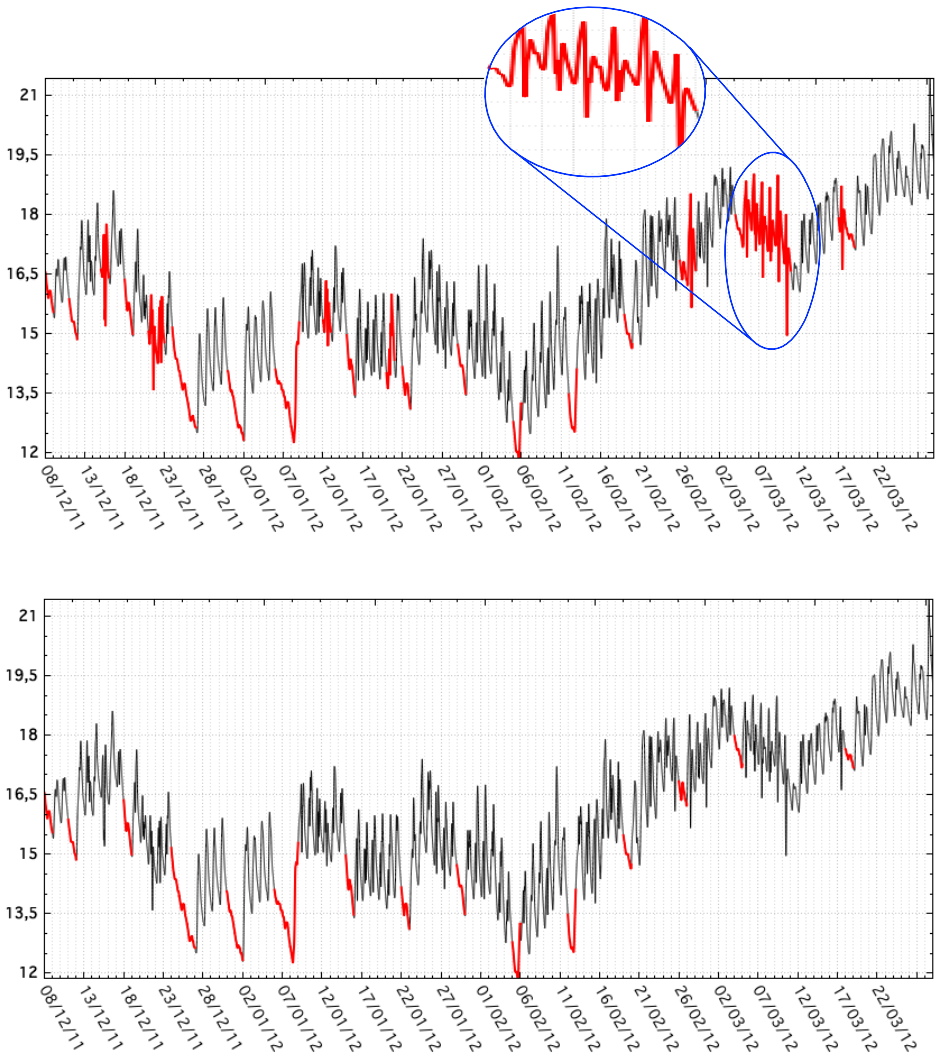

Figura 4.6: La prima immagine mostra il risultato dell'elaborazione della serie temporale di un sensore installato nell'atrio della scuola elementare Rigotti attraverso la costruzione di un modello basato su altri sensori presenti all'interno di mensa, auditorium ed uffici. Nell'immagine sotto, invece, è possibile notare il risultato che si sarebbe ottenuto analizzando il segnale solo mediante la modalità non supervisionata.

Un analisi interessante, emersa sulla base dei risultati ottenuti mediante l'utilizzo di un modello unificato, riguarda le distanze (equazione 2.6) tra i *pattern* in esso contenuto. La tabella 4.2 riporta la matrice delle distanze per le stringhe contenute all'interno del modello utilizzato nell'esempio precedente. Per semplicità i pattern vengono indicati come segue:

Pattern 1 (`bbbbcdeeffffeeeeddcccbb`)**:** modello costruito sulla base di una serie temporale relativa ad un sensore installato all'interno dell'auditorium.

Pattern 2 (`bbbbbccdeeffffeeddcccbb`)**:** modello costruito sulla base di una serie temporale relativa ad un sensore installato all'interno della mensa.

Pattern 3 (`bbbbcdeeeffffeeddccccbb`)**:** modello costruito sulla base di una serie temporale relativa ad un sensore installato all'interno di un ufficio.

	Pattern 1	**Pattern 2**	**Pattern 3**
Pattern 1	0	0.430727	0
Pattern 2	0.430727	0	0
Pattern 3	0	0	0

Tabella 4.2: Matrice delle distanze tra tutti i *pattern* contenuti nel modello unificato costruito sulla base di sensori presenti in auditorium, mensa e uffici.

Poiché tutte le distanze calcolate sono molto piccole, l'osservazione di tale analisi porterebbe a concludere che i tre *pattern* contenuti nel modello siano equivalenti e, quindi, intercambiabili. Tuttavia, la rimozione di uno di essi comporterebbe la perdita di alcune delle anomalie precedentemente individuate, come mostrato dall'immagine 4.7 nel caso della rimozione del Pattern 2. Il risultato, sicuramente sorprendente, dimostra come l'utilizzo della distanza di SAX permetta di suddividere lo spazio di ricerca in maniera differente rispetto a metriche più intuitive quali la distanza euclidea o la distanza di Hamming.

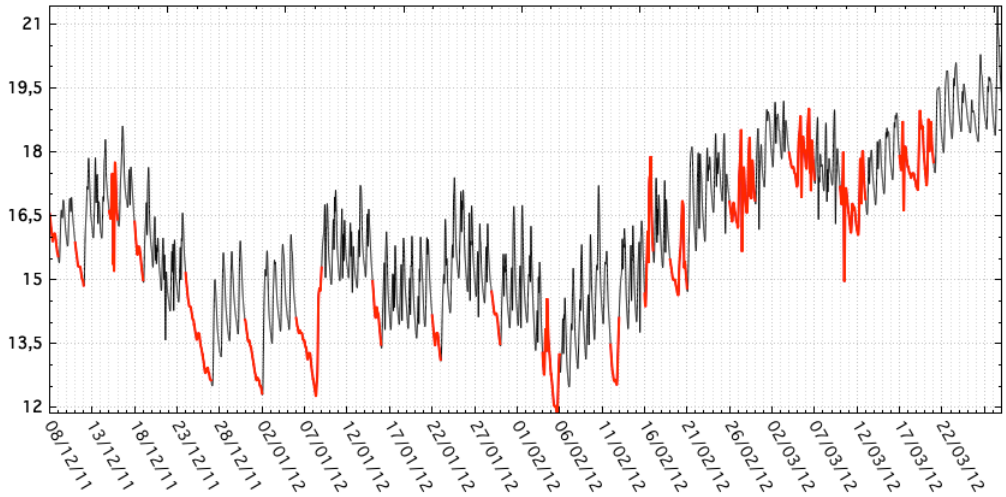

Figura 4.7: Risultato ottenuto dall'analisi mostrata in figura 4.6 utilizzando lo stesso modello dal quale è stato, però, rimosso il secondo *pattern*. Come osservabile, il numero di anomalie rilevate è decisamente inferiore.

4.2 Studio di dataset differenti

Nonostante il *tool* creato sia stato sviluppato ad *hoc* per il monitoraggio di sensori termici situati all'interno di un edificio, esso è stato messo alla prova su serie temporali differenti con l'obiettivo di testarne la capacità di generalizzazione. I *dataset* utilizzati in questa sezione sono scaricabili dal sito ufficiale di HOT SAX (`http://www.cs.ucr.edu/~eamonn/discords/`).

4.2.1 Space shuttle dataset

Questa serie temporale riguarda i dati rilevati da un sensore posto all'interno di una valvola di una navetta spaziale della NASA. Una volta individuato il periodo della serie, pari a 1000 campioni, sono state eseguite due tipologie differenti di *test*:

1. Ricerca di anomalie su *pattern* di dimensione pari al 1000 campioni. I problemi principali legati a questa scelta sono stati solamente la limitata quantità di informazione presente nel *dataset* scaricato; esso infatti conteneva solo cinque periodi. Nonostante questo `CADET` è riuscito ad individuare correttamente l'anomalia desiderata grazie ad un corretto settaggio dei parametri da esso utilizzati.

2. Ricerca di anomalie di taglia inferiore a quella del periodo (250 campioni). Tale esperimento, non previsto dall'utilizzo ufficiale di `CADET`, è stato eseguito con successo grazie alla costruzione di un modelli basato su molti *cluster* (nell'esempio in esame sono stati individuati tre centroidi differenti)

Questi esempi, che potrebbero sembrare una forzatura di applicazione per il problema originale, dimostrano come il *tool* sviluppato sia capace di rilevare diversi tipi di anomalie se supportato da una buona fase di *training* e di *parameter tuning*. La figura 4.8 mostra i risultati ottenuti.

4.2.2 Power demand dataset

Tale *dataset* contiene dati che misurano il consumo di energia di un centro di ricerca olandese per l'intero anno 1997. Esso si presta molto bene ad essere analizzato in quanto il periodo di osservazione può essere suddiviso in *frame* del tutto analoghi a quelli utilizzati sulle serie temporali della scuola Rigotti. Su tale segnale è stata effettuata un analisi di anomalie sia giornaliere che settimanali. La figura 4.9 mostra i risultati di entrambi gli esperimenti.
Come osservabile `CADET` individua correttamente tutte le anomalie, che corrispondono ai giorni festivi, e le settimane il cui numero di picchi è diverso da cinque; nel caso di analisi settimanale, tale risultato è analogo a quello fornito dagli autori di HOT SAX [23] (l'analisi giornaliera non è stata trattata nel loro studio). Questo esperimento ha dimostrato come sia possibile utilizzare il *tool* realizzato per analizzare serie temporali che contengono *pattern* di dimensione prestabilita e con andamenti riconducibili a comportamenti specifici.

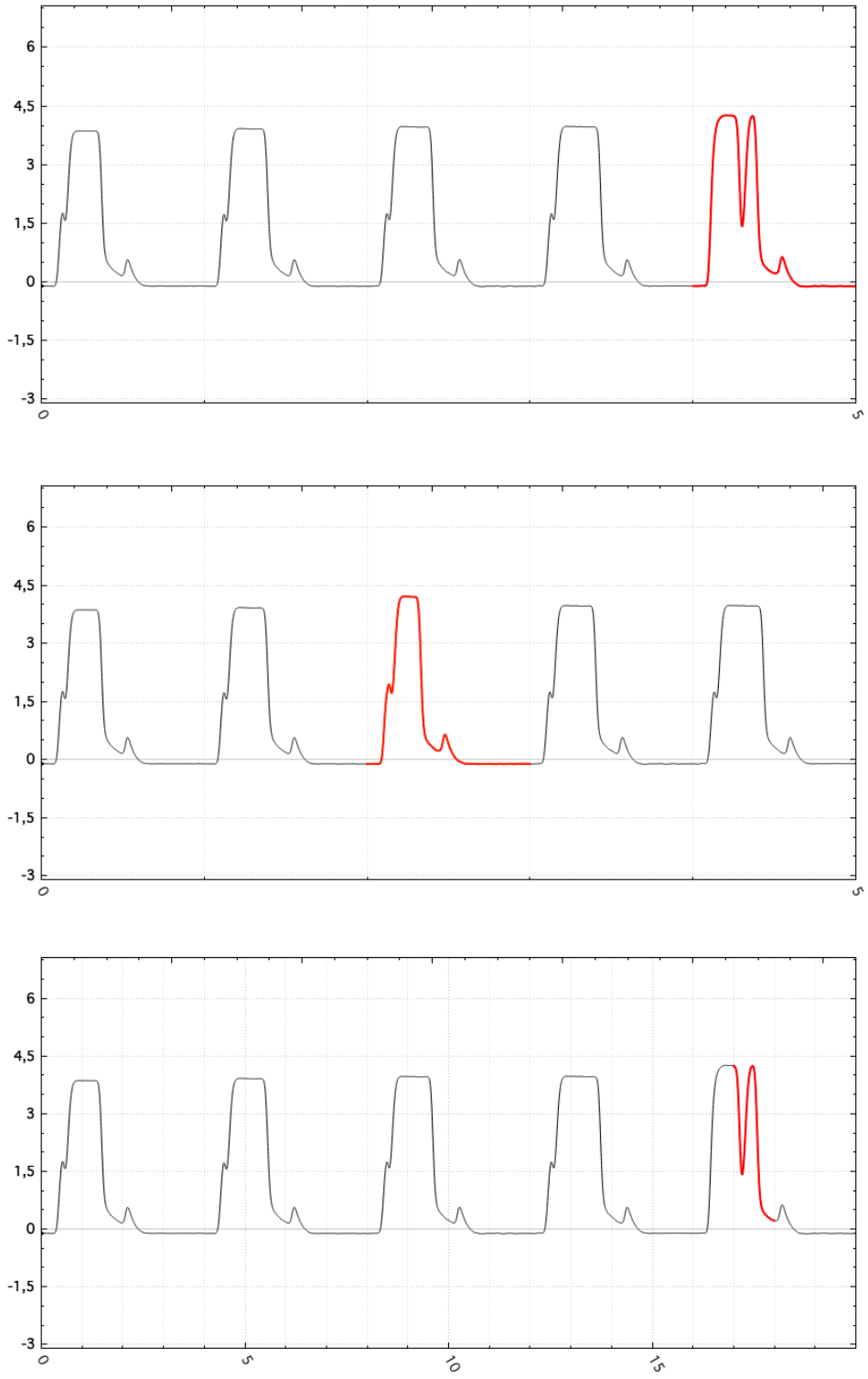

Figura 4.8: Analisi dei dati rilevati da un sensore posto all'interno di una valvola di una navetta spaziale della NASA. Nelle prime due immagini le anomalie sono state cercate analizzando i singoli periodi nella loro interezza mentre nella seconda figura le anomalie sono state ricercate come *pattern* di taglia inferiore.

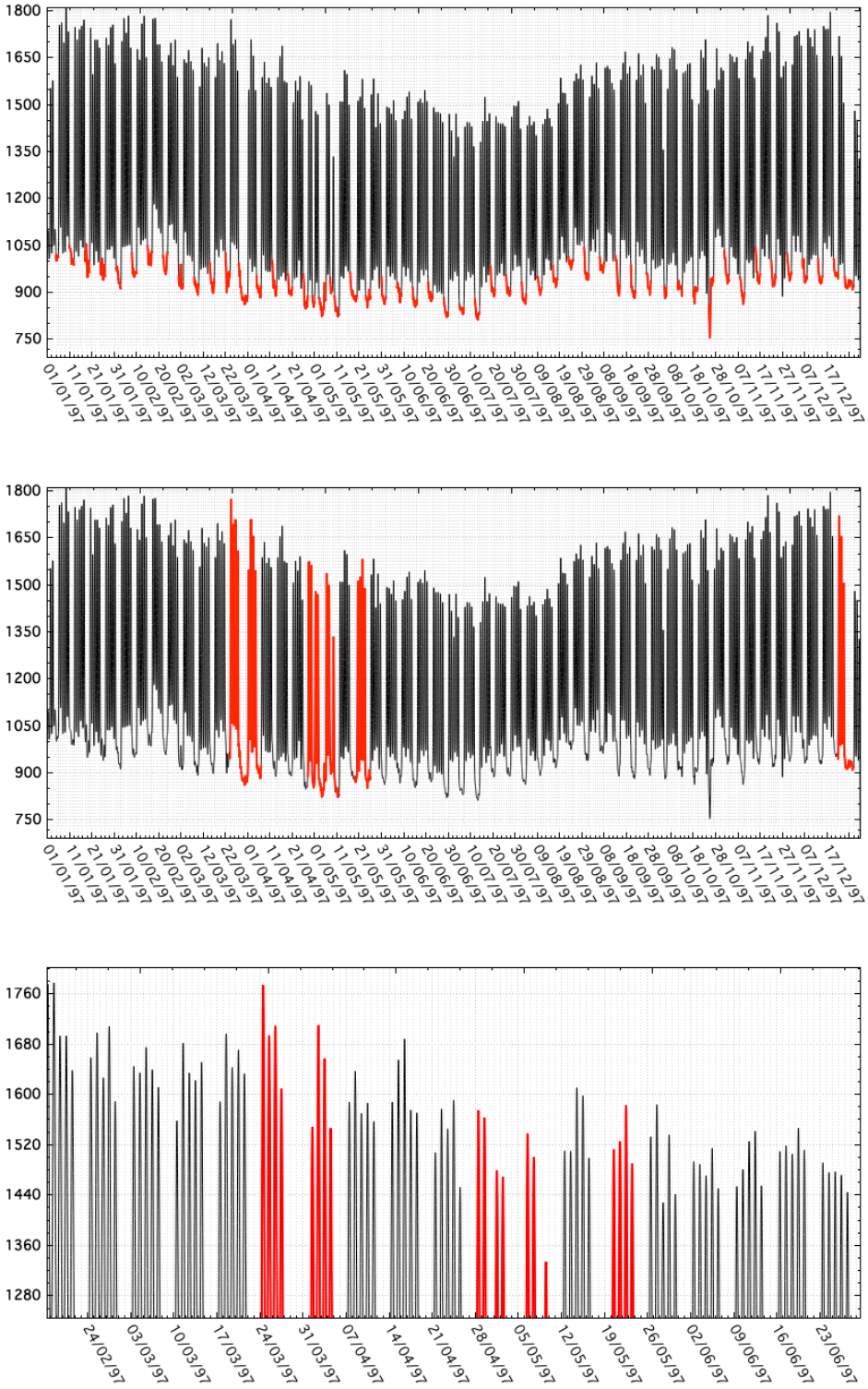

Figura 4.9: Nella prima immagine sono illustrati i risultati dell'analisi giornaliera eseguita sulla serie temporale riguardante i consumi energetici di un centro di ricerca olandese. La seconda immagine riguarda invece l'analisi delle anomalie settimanali rafforzata dall'osservazione, nell'ultima figura, dei picchi in esse presenti.

4.3 Confronto con altri tool esistenti

L'ultima serie di esperimenti eseguiti riguarda l'analisi delle serie temporali della scuola elementare Rigotti attraverso i *tool* presentati in sezione 3.1. Tali esempi non hanno la pretesa di dimostrare la superiorità di *CADET* rispetto ad altri *software* esistenti, ma ne confermano la capacità di rilevare anomalie in serie temporali di struttura specifica.

4.3.1 HOT SAX

Sebbene tale applicativo non sia reperibile in rete, è stato trovato un *tool* in Java per la gestione ed analisi di serie temporali contenente l'implementazione di HOT SAX. Il software, chiamato `JMotif` è scaricabile all'indirizzo web `https://code.google.com/p/jmotif/` ed è rilasciato secondo licenza GNU GPL v2. L'esecuzione dell'algoritmo sulle serie temporali in nostro possesso non si è dimostrata capace di rilevare le anomalie previste. È stato provato un notevole numero di combinazioni di parametri senza successo; la motivazione è probabilmente legata all'incapacità di HOT SAX di rilevare anomalie sistematiche (nel nostro *dataset* queste possono essere, ad esempio, rappresentate dai giorni festivi). Poiché l'implementazione utilizzata non è ufficiale degli autori di HOT SAX e non permette la completa libertà nel settaggio dei parametri, non è possibile esprimere un giudizio oggettivo riguardo il confronto dei risultati.

4.3.2 Assumption free anomaly detector

I problemi riscontrati nell'utilizzo di questo *tool* non sono stati condizionata principalmente dalla capacità o meno di rilevazione delle anomalie, ma dalla difficoltà di analisi dei risultati da esso forniti. Nonostante il *tool* individui un andamento del grado di anomalia che cresce attorno ai campioni ad essa corrispondenti, è difficile per un operatore umano analizzare tale funzione ed associarla ad un *pattern* specifico della serie temporale. Tale ambiguità, presente anche nei dati di esempio messi a disposizione dagli autori dell'applicativo (figura 4.10), ha reso difficile una valutazione oggettiva dei risultati forniti come *output* dell'*anomaly detector*.

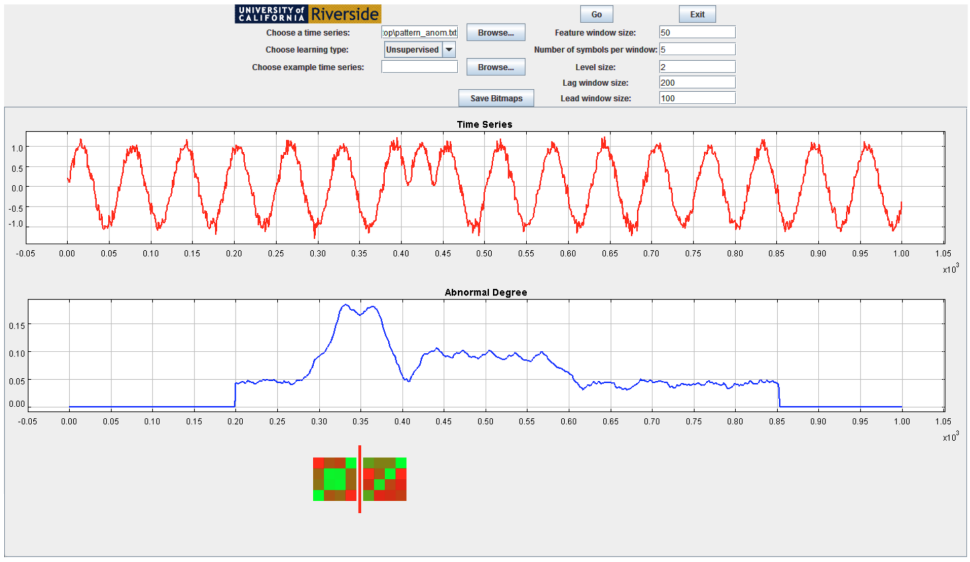

Figura 4.10: Risultato dell'elaborazione di una serie temporale di esempio, fornita sul sito ufficiale del *tool* "Assumption free anomaly detector", mediante l'utilizzo di parametri suggeriti dagli autori.

4.3.3 Analisi temporale

Il software è stato sviluppato e testato su un Macbook Pro retina (late 2013) con le seguenti specifiche:

- Sistema Operativo: Maveriks.
- Processore: 2.4 GHz Intel Core i5.
- RAM: 8GB, DDR3 @ 1600MHz.

Vengono qui riportati i dati relativi alle tempistiche delle esecuzioni di ciascuno dei tre algoritmi studiati:

CADET: circa 4 secondi per eseguire 20 iterazioni dell'algoritmo all'interno del quale la fase di *clustering* viene iterata per un massimo di 10 volte.

HOT SAX (implementazione JMotif): 145 secondi (quasi due minuti e mezzo) per l'implementazione *brute force* e 64 secondi con l'utilizzo di tecniche euristiche. Oltre alle tempistiche decisamente superiori, i risultati si sono rivelati del tutto insoddisfacenti per l'applicazione in esame.

Assumption free anomaly detector: 13 secondi. Come già discusso tale *tool* non sembra uno strumento col quale sia possibile identificare in maniera oggettiva l'anomalia rilevata.

Capitolo 5

Conclusioni e sviluppi futuri

Il lavoro di questa tesi ha affrontato il problema della rilevazione di anomalie in serie temporali attraverso un approccio basato su rappresentazione simbolica (SAX) ed una procedura iterativa capace di costruire un modello rappresentante il comportamento corretto del segnale analizzato mediante un sistema di voto. Poiché normalmente i comportamenti ritenuti accettabili di una serie temporale possono essere molteplici, tale modello può contenere uno o più *pattern* individuati attraverso l'utilizzo di un algoritmo di *clustering* sui dati originali. L'applicazione motivante è l'analisi di serie temporali di temperature interne in ambienti affollati con lo scopo specifico di rilevare anomalie che possano indicare una gestione non ottimale del sistema di riscaldamento, introducendo così il problema di analisi dei dati nel contesto di sensibilizzazione sull'utilizzo di energia e sulla progettazione di edifici intelligenti. In tali contesti, rendere gli utilizzatori consapevoli di un comportamento non corretto è il primo passo per un utilizzo più sostenibile delle risorse a disposizione. Inoltre è interessante come, dall'analisi dell'andamento di un parametro fisico ambientale, sia stato possibile ricavare informazioni riguardanti la presenza di persone all'interno delle aule monitorate. Tale risultato è particolarmente utile quando si necessita di monitorare l'occupazione di un ambiente mantenendo inviolata la *privacy* del personale in esso presente. Per validare la procedura di analisi implementata da `CADET`, sono stati condotti diversi test a livello giornaliero e settimanale sui dati messi a disposizione dalla scuola Rigotti e su altri *dataset* di dominio pubblico. I risultati, sicuramente interessanti, dimostrano come il *tool* sia stato in grado di effettuare una corretta analisi dei dati obiettivo della tesi e come, grazie alle scelte progettuali, si sia ottenuto un algoritmo generalizzabile capace di rilevare anomalie su *dataset* di varie nature. Come dimostrato, tra i pregi che caratterizzano il *tool* rispetto agli algoritmi attualmente presenti in letteratura ci sono sicuramente la velocità di rilevazione e la possibilità di esportare il modello.

Di seguito vengono elencati gli sviluppi futuri proposti per questo lavoro di tesi, alcuni dei quali sono già stati analizzati in via preliminare durante la fase implementativa di `CADET`:

Analisi di diverse metriche di distanza tra le misure di distanza più utilizzate come metriche di confronto tra stringhe è possibile esplorare l'uso della Hamming *distance*, della distanza euclidea o della *edit distance*. L'obiettivo è quello di cercare di migliorare ulteriormente la qualità del modello costruito mediante l'algoritmo di *clustering*.

Studiare un'alternativa al *substring clustering* poiché attualmente non è possibile utilizzare il *tool* per analizzare serie temporali nelle quali l'anomalia non ha una dimensione fissa (ad esempio dati riguardanti elettrocardiogrammi), si era pensato di sopperire a tale limitazione estendendo l'algoritmo con un approccio basato su finestra scorrevole. Lo studio di Keogh e Lin in [25], però, ha dimostrato come tale strada non sia percorribile. L'im-

plementazione di una procedura alternativa garantirebbe un'ulteriore generalizzazione di `CADET`.

Sostituzione dell'algoritmo k-mean è possibile provare a sostituire l'algoritmo di *k-mean clustering* con un algoritmo di tipo k-median. Quest'ultimo approccio, che ricerca i centroidi come punti appartenenti all'insieme dei dati da raggruppare, riuscirebbe a sfruttare meglio la natura discreta dei dati. L'utilizzo di *clustering* gerarchico, invece, è sconsigliata in quanto richiede di precalcolare la matrice delle distanze, operazione computazionalmente onerosa nel caso di molti punti.

Introduzione del *random restart* essendo il k-mean un algoritmo di ricerca locale, esso potrebbe fornire, come *output*, delle soluzioni che non siano minime globalmente. Una soluzione consiste nell'introduzione della tecnica di *random restart*, che prevede l'esecuzione in più iterazioni ciascuna delle quali con condizioni iniziali diverse.

Gestione automatizzata delle soglie allo stato attuale il metodo migliore per impostare le soglie utilizzate da `CADET`, in particolare quella sulla distanza tra modelli e anomalie, è quello sperimentale. Sicuramente uno studio mirato su tali valori renderebbe il *tool* autonomo in quasi ogni sua fase.

Rappresentazione del comportamento della serie mediante modelli multipli come discusso nel capitolo 3, il sistema di voto prevede l'individuazione del modello migliore tra quelli proposti durante le fasi di *clustering*. Permettere la scelta di un numero superiore di modelli potrebbe aiutare a migliorare ulteriormente la robustezza del risultato finale.

Studio di tecniche mirate per l'analisi settimanale uno possibile studio, già avviato durante la stesura di questa tesi, riguarda l'implementazione di un algoritmo mirato all'individuazione di anomalie a livello settimanale. Potrebbe risultare interessante studiare il comportamento delle varie settimane in funzione dei picchi o dei giorni anomali in esse contenuti. Tale tecnica non intaccherebbe la capacità di generalizzazione del *tool*, ma andrebbe a sfruttare in maniera più consapevole quelle che sono le caratteristiche peculiari delle serie temporali che si desidera analizzare.

Bibliografia

[1] N. A. Lynch. “Distributed Algorithms”. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1996.

[2] V. Chandola, A. Banerjee, and V. Kumar. “Anomaly detection for discrete sequences: A survey”. IEEE Trans. on Knowledge and Data Engineering, 24(5):823–839, 2012.

[3] S. Budalakoti, S. Budalakoti, A. Srivastava, M. Otey, and M. Otey. “Anomaly detection and diagnosis algorithms for discrete symbol sequences with applications to airline safety”. Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on, 39(1):101–113, Jan 2009.

[4] Q. Chen, L. Chen, X. Lian, Y. Liu, and J. X. Yu. “Indexable PLA for Efficient Similarity Search”. In VLDB, 2007.

[5] E. J. Keogh, K. Chakrabarti, S. Mehrotra, and M. J. Pazzani. “Locally Adaptive Dimensionality Reduction for Indexing Large Time Series Databases”. In SIGMOD Conference, 2001.

[6] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. “Fast Subsequence Matching in Time-Series Databases”. In SIGMOD Conference, 1994.

[7] J. Lin, E. J. Keogh, L. Wei, and S. Lonardi. “Experiencing SAX: a novel symbolic representation of time series”. Data Min. Knowl. Discov., 15(2), 2007.

[8] K. pong Chan and A. W.-C. Fu. “Efficient Time Series Matching by Wavelets”. In ICDE, 1999.

[9] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. “Fast Subsequence Matching in Time-Series Databases”. In SIGMOD Conference, 1994.

[10] F. Korn, H. V. Jagadish, and C. Faloutsos. “Efficiently supporting ad hoc queries in large datasets of time sequences”. In SIGMOD Conference, 1997.

[11] Y. Cai and R. T. Ng. “Indexing spatio-temporal trajectories with chebyshev polynomials”. In SIGMOD Conference, 2004.

[12] E. J. Keogh, K. Chakrabarti, M. J. Pazzani, and S. Mehrotra. “Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases”. Knowl. Inf. Syst., 3(3), 2001.

[13] Apostolico A, Bock ME, Lonardi S (2002). “Monotony of surprise in large-scale quest for unusual words”. In: Proceedings of the 6th International conference on research in computational molecular biology, Washington, DC, April 18–21, pp 22–31.

[14] Jin Shieh , Eamonn Keogh. “iSAX: indexing and mining terabyte sized time series”. In: Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, August 24-27, 2008. Las Vegas, Nevada, USA.

[15] Li Wei, Eamonn Keogh and Xiaopeng Xi (2006). “SAXually Explict Images: Finding Unusual Shapes”. ICDM 2006.

[16] Silvent A, Carbay C, Carry, PY, Dojat M (2003). “Data information and knowledge for medical scenario construction”. In: Proceedings of the intelligent data analysis in medicine and pharmacology workshop, Protaras, Cyprus, October 19–22.

[17] Chen JS, Moon YS, Yeung HW (2005). “Palmprint authentication using time series”. In: Proceedings of the 5th international conference on audio and video based biometric person authentication, Hilton Rye Town, NY, July 20–22.

[18] Jaya Kawale, Aditya Pal, Rob Fatland: “Tracking Spatio-Temporal Diffusion in Climate Data”. SDM 2012: 839-850.

[19] McGovern, A., Rosendahl, D. H., Kruger, A., Beaton, M. G., Brown, R. A., and Droegemeier, K. K., 2007: “Anticipating the formation of tornadoes through data mining”. Preprints, Fifth Conference on Artificial Intelligence and its Applications to Environmental Sciences, American Meteorological Society, San Antonio.

[20] Barnsley M.F. and Rising H. (1993). “Fractals Everywhere”, second edition, Academic Press.

[21] Kumar N., Lolla N., Keogh E., Lonardi S., Ratanamahatana C. and Wei L. (2005). “Time-series Bitmaps: A Practical Visualization Tool for Working with Large Time Series Databases”. SIAM 2005 Data Mining Conference.

[22] E. Keogh, L. Wei, X. Xi, S. Lonardi, J. Shieh, and S. Sirowy. “Intelligent icons: Integrating lite-weight data mining and visualization into GUI operating systems”. ICDM, 2006.

[23] Keogh E., Lin J., Fu A. (2005). “HOT SAX: efficiently finding the most unusual time series subsequence”. In: Proceedings of the 5th IEEE international conference on data mining, Houston, TX, November 27–30, pp 226–233.

[24] Wei L., Kumar N., Lolla V., Keogh E., Lonardi S., Ratanamahatana C.A. (2005). “Assumption-Free Anomaly Detection In Timeseries”. In: Proceedings of the 17th International Conference on Statistical and Scientific Database Management.

[25] E. Keogh, J. Lin, and W. Truppel. “Clustering of time series subsequences is meaningless: Implications for previous and future research”. In Proceedings of the Third IEEE International Conference on Data Mining, ICDM ’03, pages 115–, 2003.

[26] L. O’Callaghan, N. Mishra, A. Meyerson, S. Guha, and R. Motwani. “Streaming-data algorithms for high-quality clustering”. Proc. ICDE, 2002.