Indice

Capitolo 1

Serie storiche

Una serie storica è un insieme di osservazioni $\{x_t\}$, ognuna delle quali è osservata in uno specifico tempo t. Una serie storica a tempi discreti è tale che l'insieme T_0 di tempi a cui corrispondono le osservazioni fatte, è un insieme discreto. Invece le serie temporali a tempi continui sono ottenuti quando le osservazioni sono registrate continuamente su un intervallo di tempo.
Nel tentativo di eseguire una previsione dei valori futuri di una serie storica, in genere si usano solamente i valori noti (passato e presente) della serie stessa. Volendo usare solo i dati noti della serie, ci si convince che l'unica regola è ricopiare il passato, cioè si vorrebbe capire la struttura della serie nel passato al netto delle variazioni accidentali, in modo da ricopiare nel futuro solamente la struttura e non le variazioni (che non si ripeterebbero identiche). Il problema è quindi distinguere la struttura dalle fluttuazioni casuali. La struttura non è però un concetto matematicamente ben definito, è solo un'idea. Due elementi strutturali sono il trend e la periodicità (o stagionalità).

Anche questi due concetti sfuggono ad una definizione matematica univoca e precisa ma l'intuito umano li cattura ugualmente, grazie alle capacità di confronto e sintesi che abbiamo.
Un'importante parte dell'analisi delle serie storica è la selezione di un modello di probabilità adatto per i dati. Per tenere in considerazione la prossibile imprevedibilità delle osservazioni future è naturale supporre che ogni osservazione x_t è una realizzazione di una certa variabile casuale X_t

Definizione 1.1. *Un modello per le serie storiche dato un set di dati* $\{x_t\}$ *è una specificazione delle distribuzioni congiunte (o quando è possibile solo le medie e la covarianza) di una successione di variabili aleatorie* $\{X_t\}$ *i quali* $\{x_t\}$ *sono le relative realizzazioni.*

Un modello completo per le serie storiche data una successione di variabili aleatorie $\{X_1, X_2, ...\}$ specifica tutte le distribuzioni congiunte del relativo vettore aleatorio $(X_1, ..., X_n)$

$$\mathbb{P}[X_1 \leq x_1, ..., X_n \leq x_n] \quad -\infty < x_1, ..., x_n < \infty$$

Però questa specificazione è raramente usata nell'analisi delle serie storiche siccome, in generale, ci saranno molti parametri da stimare dai dati disponibili. Invece specificheremo soltanto i momenti del primo e del secondo ordine delle distribuzioni congiunte, cioè le aspettazioni $\mathbb{E}[X_t]$ e $\mathbb{E}[X_{t+h}X_t]$ con $t = 1, 2, ..., h$ e $h = 0, 1, 2, ...$, concentrandoci sulle proprietà delle X_t che dipendereanno soltando da esse. Tali proprietà di $\{X_t\}$ sono chiamate proprietà del secondo ordine. Nel caso particolare in cui le distribuzioni congiunte sono Normali multivariate, la proprietà del secondo ordine di

$\{X_t\}$ determina completamente la distribuzione congiunta e quindi ci da una completa caratterizzazione probabilistica della successione.

1.1 Alcuni modelli a media zero

Esempio 1.1 (i.i.d. noise)**.** *Il modello più semplice per una serie storica è quello che non ha componenti di trend o stagionalità e nel quale le osservazioni sono indipendenti e identicamente distribuite con media zero. Chiamiamo tale succesione di varibili aleatorie $X_1, X_2, ...$ i.i.d. noise. Per definizione possiamo scrivere, per ogni intero positivo n e per ogni realizzazione $x_1, ..., x_n$*

$$\mathbb{P}[X_1 \leq x_1, ..., X_n \leq x_n] = \mathbb{P}[X_1 \leq x_1] \dots \mathbb{P}[X_n \leq x_n] = F(x_1) \dots F(x_n)$$

dove $F(\cdot)$ è la funzione di distribuzione cumulativa di ogni variabile $X_1, X_2, ...$ In questo modello non c'è dipendenza tra le osservazioni, dunque la conoscenza di $X_1, X_2, ...X_n$ non serve per predire il comportamento di X_{n+h}, per ogni $h \geq 1$. Nonostante il modello i.i.d. noise potrebbe sembrare poco interessante per le previsioni, esso gioca un ruolo importante per i modelli di serie storiche più complessi.

Esempio 1.2 (Random walk)**.** *Il random walk $\{S_t, t = 0, 1, 2, ...\}$ è ottenuta dalla somma cumulativa di variabili aleatorie i.i.d. Un random walk di media zero è ottenuta definendo $S_0 = 0$ e*

$$S_t = X_1 + X_2 + \dots + X_t \quad t = 1, 2, ...$$

dove $\{X_t\}$ è l'i.i.d. noise.

1.2 Modelli di stagionalità e funzione di autocorrelazione

Definizione 1.2. *Una serie storica* $\{X_t, t = 0, 1, ...\}$ *è detta stazionaria se ha le stesse proprietà statistiche della serie storica traslata* $\{X_{t+h}, t = 0, 1, ...\}$ *per ogni intero* h.

Definizione 1.3. *Sia* $\{X_t\}$ *una serie storica di media* $\mathbb{E}(X_t^2) < \infty$. *Definiamo la funzione media di* $\{X_t\}$ *come*

$$\mu_X(t) = \mathbb{E}(X_t)$$

e la funzione covarianza come

$$\gamma_X(r, s) = \mathrm{Cov}(X_r, X_s) = \mathbb{E}[(X_r - \mu_X(r))(X_s - \mu_X(s))]$$

per ogni r,s *interi.*

Definizione 1.4. $\{X_t\}$ *è debolmente stazionaria se*

1. $\mu_X(t)$ *è indipendente da* t,
2. $\gamma_X(t + h, t)$ *è indipendente da* t *per ongi* h-

Remark 1

Una definzione stretta di stazionarietà di una serie storica è data dalla condizione che $(X_1, ..., X_n)$ e $(X_{1+h}, ..., X_{n+h})$ hanno la stessa distribuzione congiunta per ogni intero h e $n > 0$.

Remark 2

Per la condizione 2) della definizione di stazionarietà debole, la funzione di covarianza verrà rappresentata in termini di una variabile, in quanto

$$\gamma_X(h) = \gamma_X(h, 0) = \gamma_X(t + h, t)$$

Il termine h è chiamato ritardo.

Definizione 1.5. *Sia $\{X_t\}$ una serie storica stazionaria. La funzione di autocovarianza al ritardo h è*

$$\gamma_X(h) = \mathrm{Cov}(X_{t+h}, X_t)$$

La funzione di autocorrelazione di $\{X_t\}$ al ritardo h è

$$\rho_X(h) \equiv \frac{\gamma_X(h)}{\gamma_X(0)} = \mathrm{Cor}(X_{t+h}, X_t)$$

La funzione di covarianza gode della proprietà di linearità, cioè

$$\mathrm{Cov}(aX + bY + c, Z) = a\,\mathrm{Cov}(X, Z) + b\,\mathrm{Cov}(Y, Z)$$

Esempio 1.3 (i.i.d. noise)**.** *Se $\{X_t\}$ è i.i.d. noise e $\mathbb{E}[X_t^2] = \sigma^2 < \infty$ allora la prima richiesta della definizione di stazionarietà debole è soddisfatta in quanto $\mathbb{E}[X_t] = 0$ per ogni t. Dall'assunzione di indipendenza si ha*

$$\gamma_X(t + h, t) = \begin{cases} \sigma^2 & se \quad h = 0 \\ 0 & se \quad h \neq 0 \end{cases}$$

il quale non dipende da t. Quindi il modello i.i.d. noise è stazionario.

Esempio 1.4 (White noise)**.** *Se $\{X_t\}$ è una successione di variabili aleatorie non correlate, ognuna di media zero e varianza σ^2, allora $\{X_t\}$ è stazionaria con la stessa funzione di covarianza dell'esempio precedente.*
Chiaramente ogni successione i.i.d. noise è white noise ma non viceversa.

Esempio 1.5 (Random Walk)**.** *Se* $\{S_t\}$ *è un random walk allora* $\mathbb{E}[S_t] = 0$. $\mathbb{E}[S_t^2] = t\sigma^2$ *per ogni* t *e per ogni* $h > 0$

$$\gamma_X(t+h,t) = \mathrm{Cov}(S_{t+h}, S_t) = \mathrm{Cov}(S_t + X_{t+1} + \cdots + X_{t+h}, S_t)$$

$$= \mathrm{Cov}(S_t, S_t) = t\sigma^2$$

Siccome $\gamma_X(t+h,t)$ *dipende da* t *allora la serie* $\{S_t\}$ *non è stazionaria.*

Definizione 1.6. *Siano* $x_1, ...x_n$ *osservazioni di una serie storica. Definiamo la media campionaria di* $x_1, ...x_n$ *come*

$$\bar{x} = \frac{1}{n}\sum_{t=1}^{n} x_t$$

Definiamo la funzione di autocovarianza campionaria come

$$\hat{\gamma}(h) = n^{-1}\sum_{t=1}^{n-|h|}(x_{t+|h|} - \bar{x})(x_t - \bar{x}) \quad -n < h < n$$

Definiamo la funzione di autocorrelazione campionaria come

$$\hat{\rho}(h) = \frac{\hat{\gamma}(h)}{\hat{\gamma}(0)} \quad -n < h < n$$

Remark

Vale sempre $\hat{\rho}(0) = 1$.

Per com'è defnita, la funzione di autocorrelazione cattura le somiglianze interne alla serie storica, vede cioè se la serie traslata opportunamente somiglia a se stessa e tali somiglianze corrispondono ad aspetti strutturali. Più precisamente, se una serie ha un'accentuata periodicità ma non ha un accentuato trend, allora $\hat{\rho}(h)$ avrà un valore elevato in corrispondenza del periodo. Se invece c'è un trend accentuato, $\hat{\rho}(h)$ è abbastanza elevato dappertutto.

Capitolo 2

Dynamic Time Warping

Il Dynamic time warping, o DTW, è un algoritmo che permette l'allineamento tra due successioni (come le serie storiche), e che induce a una misura di distanza tra le due successioni allineate. Tale algoritmo è particolarmente utile per trattare successioni in cui singole componenti hanno caratteristiche che variano nel tempo e per le quali la semplice espansione, o compressione lineare delle due sequenze, non porta risultati soddisfacenti. È stato utilizzato in diversi campi di applicazione, dal riconoscimento vocale, al riconoscimento di attività motorie.

In generale, DTW è un metodo che permette di trovare una corrispondenza ottima tra due sequenze, attraverso una distorsione non lineare rispetto alla variabile indipendente (tipicamente il tempo). Alcune restrizioni per il calcolo della corrispondenza sono generalmente utilizzate: deve essere garantita la monotonicità nelle corrispondenze, ed il limite massimo di possibili corrispondenze tra elementi contigui della sequenza.

2.1 DTW classico

L'obiettivo di DTW è confrontare due successioni dipendenti dal tempo $X = (x_1, x_2, ..., x_N)$ di lunghezza $N \in \mathbb{N}$ e $Y = (y_1, y_2, ..., y_M)$ di lunghezza $M \in \mathbb{N}$. Queste successioni possono essere segnali discreti (serie storiche) o generalmente successioni di caratteristiche campionate a punti equidistanti nel tempo.

Iniziamo fissando uno spazio delle caratteriche $\mathcal{F}$. Allora $x_n, y_m \in \mathcal{F}$ con $n \in [1, N]$ e $m \in [1, M]$. Per confrontare due differenti caratteristiche $x, y \in \mathcal{F}$, c'è bisogno di una misura di costi locale, definita come

$$c : \mathcal{F} \times \mathcal{F} \longrightarrow \mathbb{R}^+$$

Generalmente $c(x, y)$ è piccola (costo basso) se x e y sono simili, altrimenti $c(x, y)$ è grande (costo alto). Valutando la misura di costi locali di ogni coppia di successioni X e Y, si ottiene ma matrice dei costi $C \in \mathbb{R}^{N \times M}$ definita come $C(n, m) = c(x_n, y_m)$. Allora l'obiettivo è di trovare un allineamento tra X e Y avendo un costo complessivo minimo.

La seguente definizione formalizza la nozione di allineamento.

Definizione 2.1. *Un (N, M)-cammino di deformazione è una successione $p = (p_1, ..., p_L)$ con $p_l = (n_l, m_l) \in [1, N] \times [1, M]$ per $l \in [1, L]$ soddisfa le seguenti condizioni*

1. *condizioni al contorno: $p_1 = (1, 1)$ e $p_l = (N, M)$*
2. *condizione di monotonicità: $n_1 \leq n_2 \leq ... \leq n_L$ e $m_1 \leq m_2 \leq ... \leq m_L$*
3. *condizione sul passo: $p_{l+1} - p_l \in \{(1, 0), (0, 1), (1, 1)\}$ per $l \in [1, L - 1]$*

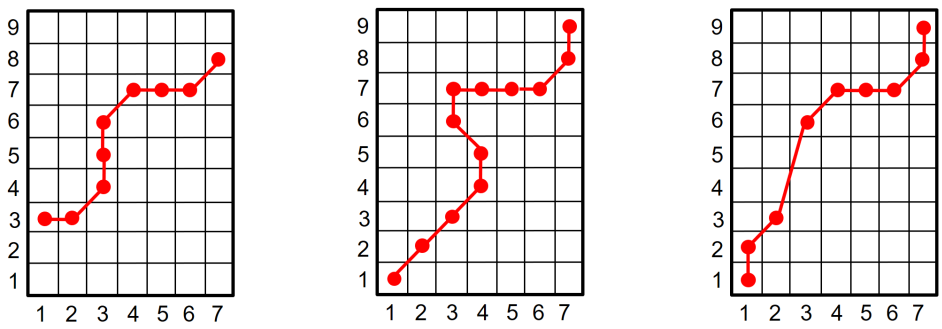

Figure 3.13 from [Müller, FMP, Springer 2015]

Figura 2.1:

Un (N, M)-cammino di deformazione $p = (p_1, .., p_L)$ definisce un allineamento tra due successioni $X = (x_1, x_2, ..., x_N)$ e $Y = (y_1, , y_2, ..., y_M)$ assegnando l'elemento x_{n_l} di X all'elemtnro y_{m_l} di Y. La condizione al contorno ci impone che i primi elementi di X e Y, così come gli ultimi elementi, sono allineati. La condizione di monotonicità riflette invece il requisito di coerenza temporale: se in X un elemento precede l'altro, allora anche i corrispondenti elementi in Y dovranno comportarsi allo stesso modo e viceversa. La condizione sul passo implica che nessun elemento di X e Y può essere omesso e che non ci sono ripetizioni nell'allineamento.

Il costo totale $c_p(X, Y)$ di un cammino di deformazione p tra X e Y in relazione al costo di misura locale c è definito come

$$c_p(X, Y) = \sum_{l=1}^{L} c(x_{n_l}, y_{m_l})$$

Inoltre un cammino di deformazione ottimale tra X e Y è un cammino di deformazione p^* avente costo totale minimo su tutti i possibili cammini di deformazione. Infine definiamo la distanza DTW tra X e Y come il costo

totale di p^*

$$\mathrm{DTW}(X,Y) = c_{p^*}(X,Y) = \min\{c_p(X,Y)| \quad p \text{ è un } (N,M) \text{ cammino di deformazione}\}$$

Remark La distanza DTW è simmetrica nel caso in cui la misura di costi locale c è simmetrica. Tuttavia la distanza DTW non è generalmente definita positiva, anche se c lo è. Per esempio otteniamo $\mathrm{DTW}(X,Y) = 0$ per la successione $X = (x_1, x_2)$ e $Y = (x_1, x_1.x_2, x_2, x_2)$ nel caso $c(x_1, x_1) = c(x_2, x_2) = 0$. Infine la distanza DTW generalmente non soddisfa la disuguaglianza triangolare anche nel caso in cui c è una metrica.

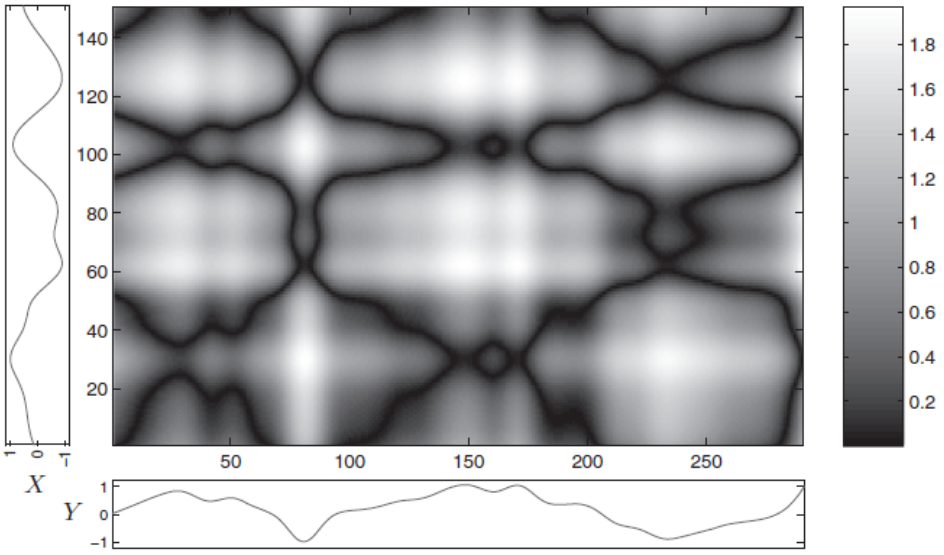

Figura 2.2: Rappresentazione matrice dei costi

Per determinare un cammino ottimale p^* si potrebbero testare tutti i possibili cammini di deformazione tra X e Y. Questo algoritmo tuttavia

è a tempo esponenziale. Vogliamo introdurre un algoritmo di complessità $\mathcal{O}(NM)$ basato sulla programmazione dinamica.
Siano $X(1:n) = (x_1, ..., x_n)$ per $n \in [1, N]$ e $Y(1, m) = (y_1, ...y_m)$ per $m \in [1 : M]$ due successioni fissate e definiamo

$$D(n, m) = \mathrm{DTW}(X(1 : n), Y(1 : m))$$

chiamata matrice dei costi accumulati. Si ha che $D(N, M) = \mathrm{DTW}(X, Y)$.Il seguente teorema ci caratterizza la matrice dei costi accumulati e ci garantisce che il costo computazione dell'algoritmo è polinomiale.

Teorema 2.1. *La matrice dei costi accumulati D soddisfa le seguenti identità*

1. *Prima riga:* $D(1, m) = \sum_{k=1}^{m} c(x_1, y_k)$
2. *Prima colonna:* $D(n, 1) = \sum_{k=1}^{m} c(x_k, y_1)$
3. *Tutti gli altri elementi:* $D(n, m) = \min\{D(n-1, m-1), D(n-1, m), D(n, m-1)\} + c(x_n, y_m)$

per $1 < n \leq N$ *e* $1 < m < M$. *In particolare* $D(N, M) = \mathrm{DTW}(X, Y)$ *può essere calcolata in* $O(MN)$ *operazioni.*

Dimostrazione. Sia $m = 1$ e $n \in [1, N]$. Allora c'è solo un possibile cammino di deformazione tra $Y(1 : 1)$ e $X(1 : n)$ avente costo totale $\sum_{k=1}^{n} c(x_k, y_1)$. Questo prova la formula per $D(n, 1)$. Allo stesso modo otteniamo la formula per $D(1, m)$. Siano $n > 1$ e $m > 1$ e sia $q = (q_1, ..., q_{L-1}, q_L)$ un cammino di deformazione ottimale tra $X(1 : n)$ e $Y(1 : m)$. Allora la condizione al contorno implica che $q_L = (n, m)$. Poniamo $(a, b) = q_{L-1}$,

allora la condizione sul passo implica che $(a, b) \in \{(n-1, m-1), (n-1, m), (n, m-1)\}$. Inoltre segue che $(q_1, ..., q_{L-1})$ deve essere il cammino di deformazione ottimale per $X(1 : a)$ e $Y(1 : b)$. Siccome $D(n, m) = c_{(q_1,...,q_{L-1})}(X(1 : a), Y(1 : b)) + c(x_n, y_m)$,(dove $c_{(q_1,...,q_{L-1})} = \min(X(1 : n), Y(1 : m)|(q_1, ..., q_{L-1})$ è un cammino di deformazione ottimale) l'ottimalità di q implica la tesi del teorema. □

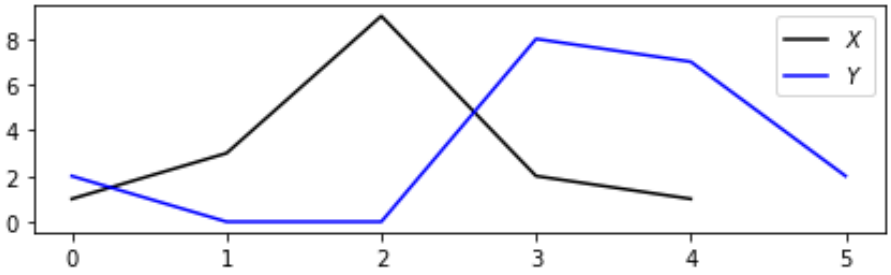

Figura 2.3: Confronto tra la successione X = (1, 3, 9, 2, 1) e Y = (2, 0, 0, 8, 7, 2)

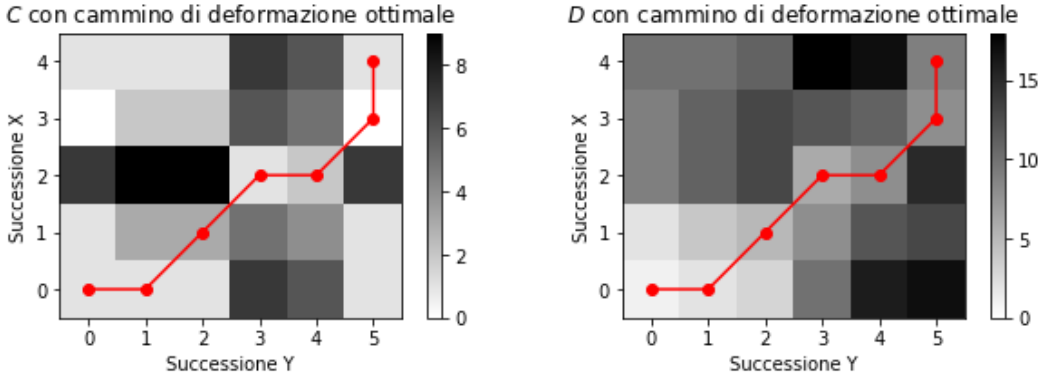

Figura 2.4: Relative matrici dei costi totali C e di costi accumulati D

2.2 Variazioni del DTW

Siamo interessati ad apportare delle modifiche all'algoritmo DTW per diminuire il costo computazione e allo stesso tempo risolvere alcuni problemi che il metodo possiede per come è stato formulato.

2.2.1 Condizione sul passo

La condizione sul passo, definita nella precedente sezione, rappresenta una condizione di continuità locale, il quale ci assicura che ogni elemento della successione X è assegnato a un elemento della successione Y e viceversa. Tuttavia un problema di questa condizione si verifica quando un singolo elemento di una successione può essere assegnato a più elementi consecutivi dell'altra successione, portando a segmenti orizzontali e verticali del cammino di deformazione. Per evitare questa degenerazione si può modificare la condizione sul passo in modo da vincolare la pendenza dei cammini di deformazione ammissibili. Come primo esempio possiamo sostituire la condizione sul passo con la condizione

$$p_{l+1} - p_l \in \{(2,1),(1,2),(1,1)\} \quad \text{per} \quad l \in [1, L]$$

Questo porta ad avere cammini di deformazione con pendenza locale limitata tra $1/2$ e 2. Di conseguenza la matrice di costi accumulati può essere calcolata attraverso la formula ricorsiva

$$D(n,m) = \min\{D(n-1,m-1), D(n-2,m), D(n,m-2)\} + c(x_n, y_m)$$

per $n \in [2,N]$ e $m \in [2,M]$. Come valori iniziali poniamo $D(0,0) = 0$, $D(1,1) = c(x_1, y_1)$, $D(n,0) = \infty$ per $n \in [1,N]$, $D(n,1) = \infty$ per $n \in [2,N]$,

$D(0,m) = \infty$ per $m \in [1,M]$, $D(1,m) = \infty$ per $m \in [2,M]$,

Notiamo che, riferendoci alla condizione sul passo modificata, esiste un cammino di deformazione rispetto a due successioni X e Y se e solo le le lunghezze N e M differiscono al più di un fattore 2. Inoltre non tutti gli elementi di X devono essere assegnati a qualche elemento di Y e viceversa. Questo permetterà di abbassare il costo computazionale. Se invece vogliamo considere tutti gli allineamenti tra gli elemeti di X e quelli di Y o viceversa, la matrice dei costi accumulati D e quindi data per ricorsione dalla seguente formula

$$D(n,m) = \min \begin{cases} D(n-1,m-1) + c(x_n, y_m) \\ D(n-2,m-1) + c(x_{n-1}, y_m) + c(x_n, y_m) \\ D(n-1,m-2) + c(x_n, y_{m-1}) + c(x_n, y_m) \\ D(n-3,m-1) + c(x_{n-2}, y_m) + c(x_{n-1}, y_m) + c(x_n, y_m) \\ D(n-1,m-3) + c(x_n, y_{m-2}) + c(x_n, y_{m-1}) + c(x_n, y_m) \end{cases}$$

per $(n,m) \in [1,N] \times [1,M]/\{(1,1)\}$.Come valori iniziali poniamo $D(1,1) = c(x_1, y_1)$, $D(n,-2) = D(n,-1) = D(n,0) = \infty$ per $n \in [2,N]$, $D(-2,m) = D(-1,m) = D(0,m) = \infty$ per $m \in [-2,M]$,

La pendenza locale del cammino di deformazione relativo varia tra $1/3$ e 3.

2.2.2 Pesi locali

Se si vuole favorire la direzione orizzontale, verticale o diagonale in un allineamento, si possono introdurre dei vettori peso $(w_d, w_h, w_v) \in \mathbb{R}^3$, dando

luogo alla ricorsione

$$D(n,m) = \min \begin{cases} D(n-1,m-1) + w_d \dot{c}(x_n, y_m) \\ D(n-1,m) + w_h \dot{c}(x_n, y_m) \\ D(n,m-1) + w_v \dot{c}(x_n, y_m) \end{cases}$$

per $n \in [2, N]$ e $m \in [2, M]$. Inoltre $D(n,1) = \sum_{k=1}^{n} w_h \dot{c}(x_n, y_1)$,$D(1,m) = sum_{k=1}^{m} w_h v\dot{c}(x_1, y_k)$ e $D(1,1) = c(x_1, y_1)$. Il caso non pesato equivale a porre $(w_d, w_h, w_v) = (1,1,1)$. Notiamo che in questo caso l'allineamento avrà una predisposizione lungo la direzione diagonale, siccome uno step diagonale corrisponde alla combinazione di uno orizzontale e uno verticale.

2.2.3 Vincoli globali

Il costo computazione dell'algoritmo DTW è $\mathcal{O}(NM)$ e l'algoritmo deve memorizzare una matrice $N \times M$. Per ridurre il costo computazione e ottimizzare la sensibilità dell'algoritmo di possono imporre dei vincoli locali. Precisamente si va a limitare ulteriormente il dominio $R \subset [1, N] \times [1, M]$ con una regione chiamata regione di vincolo. Quindi il cammino di deformazione relativo a questa regione è interamente contenuto in essa. Due esempi di regioni di vincolo sono la banda di Sakoe-Chiba e il parallelogramma di Itakura; la prima è lungo la diagonale principale e ha una larghezza fissa (orizzontale e verticale) $T \in \mathbb{N}$. Il parallelogramma di Itakura descrive una regione che vincola la pendenza a un cammino di deformazione. Precisamente, per un fissato $S \in \mathbb{R}_{>1}$, il parallelogramma di Itakura è composto da tutte celle che sono attraversate da qualche cammino di deformazione avente una pendenza

tra i valori $1/S$ e S. Tuttavia questa variazione può essere problematiche nei casi in cui il cammino ottimale nella regione di vincolo p_R^* non coincide con il cammino ottimale non vincolato p^*, e questo si verifica quando quest'ultimo non è interamente contunuto nella regione.

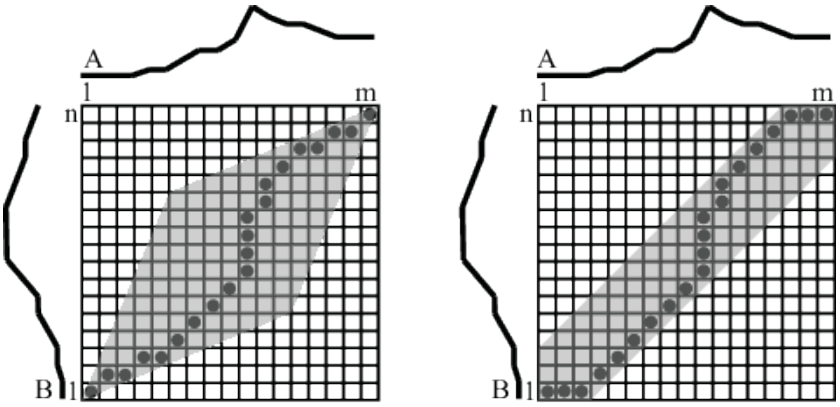

Figura 2.5: Parallelogramma di Itakura e banda di Sakoe-Chiba

Capitolo 3

Clustering

Il clustering comprende un'ampia classe di metodi atti a scoprire un numero limitato di gruppi significativi (o clusters) nei dati da sottoporre ad analisi. In particolare dato un dataset con n unità sul quale p variabili sono osservate. L'obiettivo del clustering è identificare $k(<< n)$ gruppi tali che

1. Ogni gruppo deve contenere almeno un unità
2. i gruppi sono disgiunti, cioè le unità appartengono a uno e un solo gruppo.

Ciò definisce una partizione di n unità in k cluster. Siccome si possono trovare un grande numero di partizioni, è importante spiegare il concetto di significatività. Un gruppo è significativo se le unità in questo gruppo sono simili (cioè vicine) ad altre e differenti da altre unità in altri gruppi. La similarità delle unità si verifica attraverso una opportuna metrica. I metodi standard di clustering possono essere apposimativamente distinti in due classi: i gerarchici e non gerarchici. La particolarità dei cluster gerarchici è che essi producono

una serie di partizioni ottenuti in differenti step. I cluster gerarchici richiedono quindi una matrice di distanza, a differenza dei non gerarchici. Sia X una matrice di dati di ordine $n \times p$ contenente i valori di p variabili osservate su n unità sottoposte a clustering. Il generico elemento della matrice è x_{ij} esprime il valore della variabile $j = 1, ..., p$ osservata sull'unità $i = 1, ..., n$. Le righe di X rappresentano le unità. Per ogni unità i osserviamoun vettore di valori di lunghezza p denotato come $x_i = (x_{i1}, ..., x_{ij}, ..., x_{ip})$. Se x_i e $x_{i'}$ sono vettori di variabili osservati per unità i e i' rispettivamente, possiamo considerare la distanza Euclidea:

$$d(x_i, x_{i'}) = \sqrt{\sum_{j=1}^{p} (x_{ij} - x_{i'j})^2}$$

La distanza Euclidea è un caso particolare di una classe di distanze, chiamate distanze di Minkowski di ordine q, definite come

$$d_M(x_i, x_{i'}) = \sum_{j=1}^{p} (x_{ij} - x_{i'j})^{1/q}$$

Per $q = 2$ otteniamo la distanza Euclidea, per $q = 1$ otteniamo la distanza di Manhattan.

3.1 Clustering gerarchici aglomerativi

I metodi di clustering gerarchici aglomerativi producono una serie di partizioni dove due clusters simili vengono uniti. Quindi data una matrice di distanza D_n di ordine $n \times n$, il metodo consiste nei seguenti passi:

1. si uniscono due unità/clusters che hanno la distanza minima in un nuovo cluster. Questo porta a una nuova partizione di $n - 1$ clusters;

2. Si calcola una nuova matrice di distanza D_{n-1} di ordine $((n-1) \times (n-1))$;

3. si uniscono i clusters di distanza minima utilizzando la matrice di distanza D_{n-1}, portando a una partizione di $n-2$ clusters;

4. si itera il procedimento fin quando un solo resta un solo cluster. Nell'ultimo step D_2 ha ordine 2×2.

Il punto cruciale di una procedura di clustering agglomerativa risiede nel metodo per calcolare la distanza tra un cluster e un altro. In generale differenti metodi danno differenti soluzioni. Questi metodi possono essere definiti secondo la formula di Lance-Williams. Siano C_1 e C_2 due clusters che devono essere uniti in un cluster $C_{1,2}$; la distanza tra il nuovo cluster e il cluster C_3 può essere espressa come segue:

$$d(C_{1,2}, C_3) = \alpha_1 d(C_1, C_3) + \alpha_2 d(C_2, C_3) + \beta d(C_1, C_2) + \gamma |d(C_1, C_3) - d(C_2, C_3)|$$

Quindi la classe dei metodi agglomerativi è definita dalla scelta dei parametri α_1,α_2,β e γ.

Linkage	α_1	α_2	β	γ	Alternative formula
Single	0.5	0.5	0	-0.5	$d_{ij} = \min_{x \in C_i, y \in C_j} d_{xy}$
Complete	0.5	0.5	0	0.5	$d_{ij} = \max_{x \in C_i, y \in C_j} d_{xy}$
Average	$\frac{\lvert C_i \rvert}{\lvert C_i \rvert + \lvert C_j \rvert}$	$\frac{\lvert C_j \rvert}{\lvert C_i \rvert + \lvert C_j \rvert}$	0	0	$d_{ij} = \frac{1}{\lvert C_i \rvert \lvert C_j \rvert} \sum_{x \in C_i, y \in C_j} d_{xy}$
Weighted	0.5	0.5	0	0	
Centroid	$\frac{\lvert C_i \rvert}{\lvert C_i \rvert + \lvert C_j \rvert}$	$\frac{\lvert C_j \rvert}{\lvert C_i \rvert + \lvert C_j \rvert}$	$-\frac{\lvert C_i \rvert \lvert C_j \rvert}{(\lvert C_i \rvert + \lvert C_j \rvert)^2}$	0	
Median	0.5	0.5	-0.25	0	
Ward	$\frac{\lvert C_i \rvert + \lvert C_m \rvert}{\lvert C_i \rvert + \lvert C_j \rvert + \lvert C_m \rvert}$	$\frac{\lvert C_j \rvert + \lvert C_m \rvert}{\lvert C_i \rvert + \lvert C_j \rvert + \lvert C_m \rvert}$	$-\frac{\lvert C_m \rvert}{\lvert C_i \rvert + \lvert C_j \rvert + \lvert C_m \rvert}$	0	

Figura 3.1: Formula di Lance-Williams: alcuni casi particolari

3.2 Clustering gerarchici divisivi

I metodi di clustering gerarchici divisivi fanno parte di una classe di metodi di tipo "top down" (dall'alto verso il basso) in cui tutti gli elementi si trovano inizialmente in un singolo cluster che viene via via suddiviso ricorsivamente in sotto-cluster. I clustering divisivi sono generalmente meno comuni di quelli accumulativi, questo per una mera questione computazionale. Infatti siccome le procedure di clustering divisivi separano un cluster esistente in due clusters, il problema è come trovare la separazione ottimale. Una valida soluzione è stata data da una procedura di clustering divisiva famosa, il metodo DIANA (DIvisive ANAlysis Clustering): al primo step per dividere il cluster iniziale, DIANA cerca l'unità con la più grande distanza media dalle altre unità. Tale unità e dunque usata per costruire un nuovo cluster che viene chiamato separatore. Se alcune unità sono vicine al separatore, allora esse appartengono al suo gruppo. Questo procedimento si ripete fin quando tutte le unità si sono spostate. Si procede poi nello stesso modo ma con il cluster originale diviso in due cluster, che si separeranno in quattro cluster ecc.

3.3 Clustering non gerarchici

Siamo interessati a studiare i metodi di clustering non gerarchici. Può risultare contro intuitivo utilizzare questa classe di metodi rispetto ai gerarchici; infatti quest'ultimi sono molto più semplici da implementare, siccome i non genrarchici richiedono la conoscenza preliminare del numero di clusters anche

quando la conoscenza preliminare dei dati non è disponibile. L'uso di questi metodi però è giustificato dalle seguenti proprietà:

1. I metodi non gerarchici non richiedono il calcolo di una matrice di distanze. Questo è particolarmente rilevante per dataset molto grandi per il quale il calcolo della matrice di distanze ha costo alto.

2. Nei metodi di clustering gerarchici agglomerativi quando due unità appartengono allo stesso cluster, esse non possono essere separate al passo successivo, oppure, in quelli divisivi, se le unità sono divise non possono essere più unite al passo successivo.

3.3.1 k-Means

L'algoritmo k-Means è uno dei più famosi metodi di clustering non gerarchici. Esso può essere applicato quando le osservazioni sono quantitative. Esso cerca la miglior partizione di n unità in k clusters. Per capire meglio il concetto di miglior partizione, è utile decomporre la somma totale dei quadrati in due termini

$$T = W + B$$

dove

- $T = \sum_{i=1}^{n} \sum_{j=1}^{p} (x_{ij} - \bar{x}_j)^2$
- $W = \sum_{g=1}^{k} W_g$ con $W_g = \sum_{i=1}^{n_g} \sum_{j=1}^{p} (x_{ij} - \bar{x}_{gj})^2$
- $B = \sum_{g=1}^{k} n_g (\bar{x}_{gj} - \bar{x}_j)^2$

dove n_g è il numero di unità apparteneti al cluster g, $\bar{x}_j = \frac{\sum_{i=1}^{n} x_{ij}}{n}$ la media complessiva della j-esima variabile e $\bar{x}_{gj} = \frac{\sum_{i=1}^{n_g} x_{ij}}{n_g}$ il valore medio della j-esima variabile nel g-esimo cluster.

La quantità W permette di valutare la qualità di una partizione; infatti esso è espresso come somma di W_g, dove W_g è una misura della variabilità delle osservazioni all'interno di ciascun cluster. Se tutte le unità hanno lo stesso valore, allora $W_g = 0$.

Segue che la miglior partizione di n unità in k clusters può essere definita come

$$\min W$$

o equivalentemente $\max B$.

W può essere espressa in termini di una matrice U di ordine $(n \times k)$. Essa è una matrice binaria tale che ogni riga corrisponde a un'unità e contiene solo un elemento uguale a 1. Questo elemento denota l'appartenza dell'unità al cluster. Denotiamo inoltre con H la matrice dei centroidi di ordine $(k \times p)$ di righe $h_g = (h_{g1}, ..., h_{gp})$, $g = 1, ..., k$. La soluzione ottimale può essere trovata tramite il seguente algoritmo:

1. Razionalmente o casulamente scegliamo k centroidi iniziali, cioè la matrice dei centroidi H

2. Data H assegnamo ogni unità a un cluster in modo che la distanza dell'unità dal centroide sia minima:

$$u_{ig} = \begin{cases} 1 & \text{se} \quad g = \underset{g'=1,...,k}{\arg\min}\ d^2(x_i, h_{g'}) \\ 0 & \text{altrimenti} \end{cases}$$

3. Data U, calcoliamo i centroidi

$$h_g = \frac{\sum_{i=1}^{n} u_{ig} x_i}{\sum_{i=1}^{n} u - ig}$$

con $g = 1, ..., k$

4. Si ripetano i passi 2 e 3 fin quando non ci sono più modifiche in due iterazioni consecutive.

3.3.2 k-Medoids

Un altro famoso algoritmo di clustering non gerarchico è il k-Medoids. La più grande differenza con l'algoritmo k-Means è che i clusters sono interpretati in termini di modoidi, un sottinsieme di osservabili che caratterizzano i clusters. Dunque i prototipi di cluster non sono più entità fittizie come i centroidi, calcolati come medie di unità assegnate ai cluster, ma reali entità osservate. L'algoritmo per la soluzione ottimale è simile a quello del k-Means:

1. Razionalmente o casulamente scegliamo k medoidi iniziali, cioè la matrice dei medoidi H

2. Data H assegnamo ogni unità a un cluster in modo che la sua distanza dal medoide è minima:

$$u_{ig} = \begin{cases} 1 \quad \text{se} \quad g = \underset{g'=1,...,k}{\arg\min}\ d^2(x_i, h_{g'}) \\ \\ 0 \quad \text{altrimenti} \end{cases}$$

3. Data U, calcoliamo i medoidi

$$h_g = \underset{i=1,..,n}{\arg\min} \sum_{i'=1}^{n} d^2(x_i, x_{i'})$$

con $g = 1, ..., k$

4. Si ripetano i passi 2 e 3 fin quando non ci sono più modifiche in due iterazioni consecutive.

3.4 k-Nearest Neighbors

Nel problema delle classificazioni esistono due tipo di scenari che gli statistici possono riscontrare:

- si ha conoscenza della distribuzione congiunta di osservazioni x e della categoria C, quindi in questo caso si applica l'analisi di Bayes;
- non si ha conoscenza della distribuzione congiunta, la decisione di classificare x nella categoria C può dipendere solo da una collezione di n campioni $(x_1, C_1), (x_2, C_2), ..., (x_n, C_n)$ e il processo di decisione non è chiaro.

L'ultimo problema rientra nell'ambito della statistica non parametriche. Se supponiamo che i campioni classificati (x_i, C_i) sono i.i.d. con la stessa distribuzione di (x, C), può essere ragionevole assumere che le osservazioni vicine tra di loro (in una metrica appropriata) avranno la stessa classificazione. La più semplice procedura di classificazione è il nearest neighbor (NN) che classifica x nella categoria del suo vicino. Lo scopo di questa sezione è illustrare il metodo e mostrare che la probabilità di errore di esso è più piccola della probabilità di errore di Bayes, e che quindi è più piccola di ogni altro metodo di decisione.

Sia $(x_1, C_1), (x_2, C_2), ..., (x_n, C_n)$ un insieme di n coppie, dove x_i hanno valori in uno spazio metrico dove è definita una metrica d, e C_i prende valori in un insieme discreto $\{1, 2, ..., M\}$. Sia (x, C) una coppia, dove solo la misurazione x è osservabile e si desidera stimare C utilizzando le informazioni contenute in un insieme di punti classificati correttamente. Diremo che

$$x'_n \in \{x_1, ..., x_n\}$$

è il vicino più prossimo di x se

$$\min d(x_i, x) = d(x'_n, x) \quad i = 1, 2, ..., n$$

Questa regola stabilisce che x appartiene alla categoria C'_n del suo vicino più prossimo x'_n.

Se il numero dei campioni è grande, piuttosto che considerare il singolo vicino più prossimo, si considerano la maggior parte dei k-vicini più prossimi. Per l'analisi dell'errore ci rifaremo al caso in cui $k = 1$, cioè il caso ottimale.

Una proprietà interessante di questo metodo riguarda la convergenza.

Lemma 3.1. *Siano x e $x_1, ..., x_n$ variabili aleatorie i.i.d a valori in uno spazio metrico X. Denotiamo con $x'_n \in \{x_1, ..., x_n\}$ il vicino più prossimo a x. Allora $x'_n \longrightarrow x$ con probabilità 1.*

Teorema 3.1. *Sia X uno spazio metrico separabile. Siano f_1 e f_2 tali che, con probabilità 1, x è punto di continuità di f_1 e f_2, oppure un punto di probabilità non nulla. Allora il rischio R (la probabilità di errore) ha bounds*

$$R^* \leq R \leq 2R^*(1 - R^*)$$

Dove R^* è il rischio del metodo di Bayes.

Capitolo 4

Sperimentazione

E' stato implementato tramite il cluster sulle serie temporali con la distanza DTW. In mancanza di dataset sono stati simulati le serie temporali attraverso 100 campionamenti di moti browniani e 100 campionamenti di moti browniani geometrici.

```r
if (!require("class")) install.packages("class")
if (!require("dtw")) install.packages("dtw")
library(dtw)
ntrial=50
ntest=50
tslength=50
deltat=1
d=1
testdataset1<-matrix(ncol=tslength, nrow=ntest)
testdataset2<-matrix(ncol=tslength, nrow=ntest)
traindataset1<-matrix(ncol=tslength, nrow=ntrial)
traindataset2<-matrix(ncol=tslength, nrow=ntrial)
for(i in c(1:ntest)){
  testdataset1[1,i]=0
  testdataset2[1,i]=0
  for (j in c(2:tslength)){
    testdataset1[j,i]=testdataset1[j-1,i]+sqrt(deltat)*rnorm(1)
    testdataset2[j,i]=testdataset2[j-1,i]+d*deltat+sqrt(deltat)*rnorm(1)
  }
}

for (i in c(1:ntrial)){
  traindataset1[1,i]=0
  traindataset2[1,i]=0
  for (j in c(2:tslength)){
    traindataset1[j,i]=traindataset1[j-1,i]+rnorm(1)
    traindataset2[j,i]=traindataset2[j-1,i]+0.5+rnorm(1)
  }
}
testdataset<-rbind(traindataset1,traindataset2,testdataset1,testdataset2)
train.labels <- c(rep(1, ntrial), rep(2, ntrial))
dist1 <- dist(testdataset, method = "dtw")
dist.matrix <- as.matrix(dist1)
```

```r
k<-3
indnn<-matrix(ncol=k, nrow=(2*ntest))
for (i in c(1:(2*ntest))){
  disttest<-dist.matrix[1:(2*ntrial),i+(2*ntrial)]
  sortedist<-sort(disttest)
  indnn1<-which(disttest %in% sortedist[1:k])
  if (length(indnn1)!=k){
    print("elementi con la stessa distanza: effettuato troncamento")
  }
  indnn[i,]<-indnn1[1:k]
}
group<-matrix(nrow=2*ntest, ncol=k)
for (i in c(1:(2*ntest))){
  for (j in c(1:k)){
    if (indnn[i,j]<=ntrial){
      group[i,j]=1
    }else{
      group[i,j]=2
    }
  }
}
finalgroup<-vector(length=2*ntest)
for (i in c(1:(2*ntest))){
  groupings <- unique(group[i,])
  finalgroup[i]=groupings[which.max(tabulate(match(group[i,], groupings)))]
}

finalgroup
```

Fisssando il numero di classi $k = 2$ si è proceduto andando a costruire

la matrici delle distanze, troncando gli elementi con la stessa distanza. In seguito (siccome l'algoritmo del knn di R non matcha con questra distanza) si è fatto labeling e creati i cluster.

```
63  plot.df <- data.frame(traindataset2,testdataset2, finalgroup)
64  plot(1:t, plot.df$X1, col=c("red","blue")[finalgroup], xlab="tempo", ylab="")
65  points(1:t, plot.df$X2, col=c("red","blue")[finalgroup])
66  points(1:t, plot.df$X3, col=c("red","blue")[finalgroup])
67  points(1:t, plot.df$X4, col=c("red","blue")[finalgroup])
68  points(1:t, plot.df$X5, col=c("red","blue")[finalgroup])
69  points(1:t, plot.df$X6, col=c("red","blue")[finalgroup])
70  points(1:t, plot.df$X7, col=c("red","blue")[finalgroup])
71  points(1:t, plot.df$X8, col=c("red","blue")[finalgroup])
72  points(1:t, plot.df$X9, col=c("red","blue")[finalgroup])
73  points(1:t, plot.df$X10, col=c("red","blue")[finalgroup])
```

Infine si è plottato il risultato. Le due classi sono separate molto bene, eccetto quanche punto che va in overfitting. In generale con questa semplice simulazione si è riuscito ad avere un buon risulato.

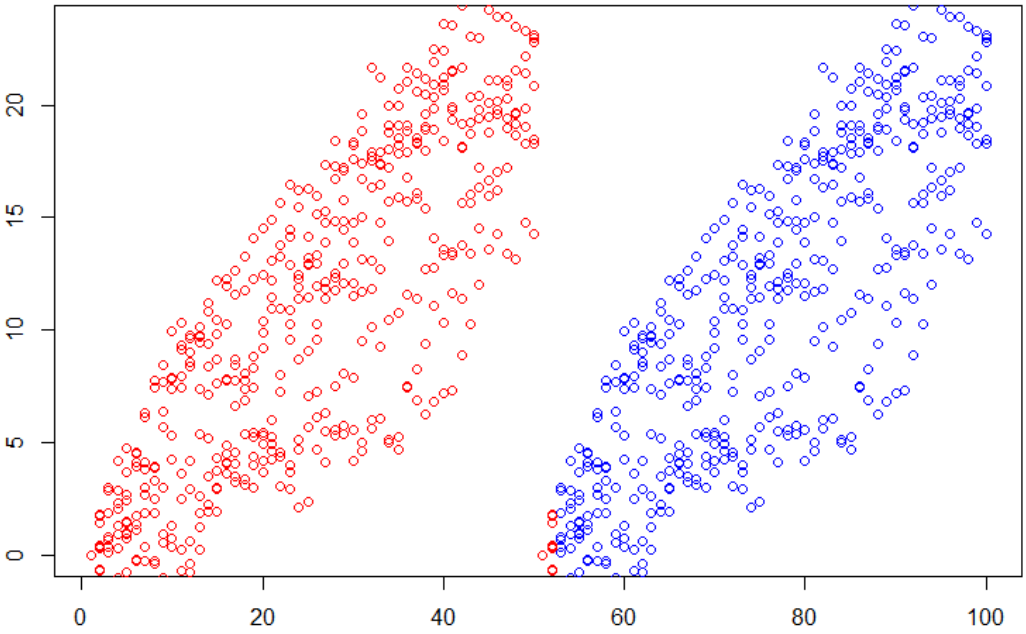

Figura 4.1: Plot clustering sulle serie storiche $k = 2$

Bibliografia

[1] Peter J. Brockwell, Richard A. Davis, *Introduction to Time Series and Forecasting*, Springer Cham

[2] T. Cover and P. Hart, *Nearest neighbor pattern classification* in IEEE Transactions on Information Theory, vol. 13, no. 1, pp. 21-27, January 1967, doi: 10.1109/TIT.1967.1053964.

[3] Franco Flandoli *Dispense di Statistica II*, 2013-2014

[4] Paolo Giordani, Maria Brigida Ferraro, Francesca Martella, *An Introduction to Clustering with R*, Springer Singapore

[5] Meinard Müller, *Information Retrieval for Music and Motion*, Springer-Verlag Berlin Heidelberg 2007

[6] Olga Valenzuela, Fernando Rojas, Luis Javier Herrera, Héctor Pomares, Ignacio Rojas, *Theory and Applications of Time Series Analysis and Forecasting*, Selected Contributions from ITISE 2021.