UNIVERSITÀ DEGLI STUDI DI MILANO

Dipartimento di Matematica

Dottorato di Ricerca in Matematica e Statistica per le Scienze Computazionali

XXII Ciclo

Candidato: Dott. Francesco Saccà

Problemi di Clustering con vincoli: algoritmi e complessità.

Relatori: Prof. A. Bertoni, Prof. G. Valentini

Anno Accademico 2009-2010

Indice

Ringraziamenti

Si ringrazia il Dott. Matteo Re per aver segnalato possibili applicazioni del Clustering vincolato in R^1 al problema dell'identificazione di regioni promotore in sequenze genomiche.

Introduzione

Un compito di rilievo nella elaborazione dati è quello di classificare i dati in base a qualche criterio, cioè suddividere un insieme di dati grezzi in sottoinsiemi 'significativi', detti appunto '**cluster**', i quali risultino essere simultaneamente omogenei e ben separati: l'omogeneità implica che all'interno dello stesso cluster le entità debbano essere rassomiglianti, mentre la separabilità impone la difformità tra entità appartenenti a cluster diversi.
L'importanza di questa problematica, che chiameremo 'Problema di Clustering', è dimostrata dalla quantità di lavori sia teorici che sperimentali in questo ambito; per una rassegna si veda ad esempio [Jain Murty Flynn 1999].

Per quanto riguarda il presente lavoro, ci si limiterà a considerare come clustering il seguente problema: dati n elementi $x_1, x_2, ..., x_n \in R^N$ e un intero $m > 1$, si richiede di determinare una partizione $(A_1, A_2, ..., A_m)$ di $\{1, 2, ..., n\}$ in m classi che minimizzi la funzione obiettivo

$$\sum_{i=1}^{m} \sum_{j \in A_i} ||x_j - C_{A_i}||_p^p$$

dove C_{A_i} è il p-centroide di A_i rispetto alla norma $||.||_p$.
Il caso $N = 1$ presenta una proprietà peculiare detta **'String Property'**: la partizione ottima $(A_1, A_2, ..., A_m)$ è tale che, per ogni classe A_j, i numeri $\{x_i | i \in A_j\}$ sono consecutivi nella sequenza $x_1 < x_2 < ... < x_n$ (si dirà che la partizione è contigua). Tale proprietà è stata dimostrata per la prima volta da

Edwards e Cavalli-Sforza (1965) [Edwards e Cavalli-Sforza 1965] per il clustering in R^1 con norma L_2, ed è stata estesa da Novick (2009) [Novick 2009] al clustering in R^1 con ogni norma L_p.

In questa Tesi introduciamo, e studiamo nel caso $N = 1$, il problema che chiameremo di 'Clustering vincolato', imponendo a priori dei vincoli sulle cardinalità delle singole classi delle partizioni $(A_1, A_2, ..., A_m)$.
Dopo aver introdotto nel **Capitolo 1** le principali nozioni sul Clustering, unitamente ad una panoramica sui metodi più noti, nel **Capitolo 2** invece esamineremo, data una qualunque norma $||.||_p$, un sottoproblema preliminare al Clustering (e al Clustering vincolato), che è quello della Localizzazione del p-Centroide di un singolo cluster: dati i numeri razionali $x_1 \leq x_2 \leq .... \leq x_n$ ed un numero razionale r, decidere se $C_p < r$ dove:

$$C_p = arg\min_x \sum_{i=1}^{n} |x_i - x|^p$$

Tale problema, nella sua apparente semplicità, non sembra banale dal punto di vista della complessità computazionale: infatti, proveremo che il problema SQRT-Sum è polinomialmente riducibile alla Localizzazione del $\frac{3}{2}$-Centroide. SQRT-Sum richiede, dati i razionali $k_1 \leq k_2 \leq .... \leq k_n$ ed un numero razionale a, di decidere se

$$\sqrt{k_1} + ... + \sqrt{k_m} > a.$$

Questo problema venne introdotto in modo naturale nel 1976 da Garey e Johnson [Garey Graham Johnson 1976], in relazione al Problema del Commesso Viaggiatore Euclideo e, a dispetto di molti sforzi finalizzati a determinarne la complessità, il miglior risultato pone attualmente SQRT-Sum in **CH** [ABKM 2006], la classe di complessità introdotta nel 1986 da Wagner [Wagner 1986]. Pertanto, la scoperta di tale riduzione polinomiale pone al

problema della Localizzazione del p-Centroide una limitazione inferiore di complessità difficilmente migliorabile.
Il principale risultato ottenuto in questo capitolo è il **Teorema 2.1:**

1. Per ogni p razionale, il Problema della Localizzazione del p-Centroide $\in$ **CH**;

2. Se p intero, il Problema della Localizzazione del p-Centroide $\in$ **P**.

Nel **Capitolo 3**, dopo aver introdotto formalmente il problema di Clustering vincolato, vengono studiate le proprietà delle soluzioni ottime nel caso in cui gli elementi da clusterizzare siano numeri razionali (Clustering vincolato in R^1). Il principale contributo in questo capitolo è quello di estendere la 'String Property' (provata per clustering in R^1 con ogni norma L_p in [Novick 2009]) al Clustering vincolato (**Teorema 3.1** e **Teorema 3.3**).

Nel **Capitolo 4** useremo la 'String Property' (precedentemente dimostrata) per lo sviluppo di algoritmi particolarmente efficienti per vari problemi di Clustering vincolato in R^1: infatti, limitare lo spazio di ricerca solo a partizioni contigue introduce forti semplificazioni al problema, semplificazioni algoritmicamente vantaggiose.
Per prima cosa viene presentato un algoritmo efficiente per il bi-clustering vincolato (**Alg BCV**): la naturale estensione al caso generale (partizioni in m classi anzicchè in *2*) dell'algoritmo del bi-clustering porta ad algoritmi che lavorano in tempo esponenziale. Successivamente, viene provata l'impossibilità di trovare per tale problema algoritmi che lavorino in tempo polinomiale dimostrando che, per ogni norma $||.||_p$, il problema del Clustering vincolato in R^1 è **NP**-difficile, e diventa **NP**-completo nel caso di $||.||_1$ o $||.||_2$ (**Teorema 4.1**).

Viene poi studiata una versione rilassata del problema del Clustering vincolato, in cui si richiede che le cardinalità delle classi appartengano ad un insieme numerico predefinito. Per tale problema in R^1 si è riusciti a realizzare un efficiente algoritmo basato su tecniche di programmazione dinamica (**mclusterdinamico**).

Infine, viene presentata una potenziale applicazione di una forma ottimizzata di questo algoritmo (**mclusterdinamico2**) ad un problema di ambito bioinformatico: in particolare si osserva che, a un certo livello di approssimazione, l'identificazione di regioni promotore in sequenze genomiche può essere modellata come un problema di clustering con vincoli in R^1, in maniera analoga a quanto trattato da [Shmid 2007]. I questa Tesi i due metodi vengono confrontati e viene discussa la rilevazione di nuovi geni, insieme alla presenza di falsi positivi.

Capitolo 1

Clustering

1.1 Introduzione

1.1.1 Finalità

Il Clustering consiste nella classificazione non supervisionata di dati in gruppi (cluster).
Nell'apprendimento non supervisionato non si dispone di etichette associabili ai dati da classificare e nel caso più generale non è noto a priori nemmeno il numero delle classi (cluster) presenti nei dati.
La mancanza di una 'ground truth' obiettiva ed osservabile tramite cui verificare l'accuratezza e l'attendibilità delle partizioni ottenute da algoritmi di clustering, rende tale problema di apprendimento particolarmente complesso, sostanzialmente 'mal posto' [Ben-David 2009].
L'indisponibilità di conoscenza a priori sui dati (in forma di etichette) delle classi rende necessario lo sviluppo di algoritmi basati sulla analisi delle sole caratteristiche (feature) dei dati per la ricerca di strutture presenti nei dati stessi.

A tal fine, gli algoritmi di clustering ricercano raggruppamenti di elementi dei dati (cluster) sulla base di opportune misure di similarità/distanza definite rispetto alle feature caratterizzanti i dati stessi.
Informalmente, l'obiettivo del clustering consiste nell'individuare raggruppamenti simultaneamente omogenei e ben separati: l'omogeneità implica che all'interno dello stesso cluster le entità debbano essere rassomiglianti, mentre la separabilità impone la difformità tra entità appartenenti a cluster diversi. In altre parole, un elemento appartenente a un cluster dovrà essere 'simile' a tutti gli altri elementi del medesimo cluster, mentre dovrà essere sostanzialmente 'diverso' da quelli presenti in tutti gli altri cluster.
Pertanto, il **Clustering** o **Analisi dei Cluster** (dal termine inglese Cluster Analysis introdotto da Robert Tryon nel 1939) è un insieme di tecniche di analisi multivariata dei dati volte alla selezione e raggruppamento di elementi omogenei in un insieme di dati. Esso è da tempo oggetto di specifiche ricerche, grazie alle quali sono nati diversi approcci al problema. Studi recenti hanno da un lato lavorato per unificare alcune classi di algoritmi apparentemente diversi tra loro e dall'altro prodotto tecniche [Ben-David 2006, Ben-David 2008] per affrontare più efficacemente problemi che coinvolgono dati sparsi, di grandi dimensioni o provenienti da particolari applicazioni (con specifiche peculiarità ed esigenze), quali quelle in campo bioinformatico.
Più precisamente, i sistemi di classificazione si distinguono in **supervisionati** o **non supervisionati** [Duda et al. 2001]. Gli algoritmi **supervisionati** assegnano i dati di input a un numero finito di classi apprendendo una funzione di classificazione da esempi etichettati (cioè la cui appartenenza al 'giusto' cluster è nota a priori), minimizzando una funzione di costo opportunamente stabilita. Naturalmente, e questo è il caso maggiormente frequente

in pratica, non sempre esempi etichettati sono disponibili: in tal caso, lo scopo degli algoritmi **non supervisionati** (anche detti di '**Clustering**') è assegnare i dati di input in un numero finito (e in generale incognito) di 'naturali' e 'nascoste' sottoclassi (cluster), a patto di effettuare una classificazione di esempi ignoti ma generati da una medesima distribuzione di probalilità [Baraldi e Alpaydin 2002]. Inoltre, uno dei possibili ed alternativi usi degli algoritmi di Clustering è quello di ottenere una rappresentazione compressa di dati di grandi dimensioni.
Più formalmente, come vedremo in seguito, il **Clustering** è la forma più comune di **apprendimento non supervisionato** [MRS 2007]. L'assenza di supervisione significa che non viene eseguito alcun processo di assegnazione manuale di oggetti alle classi per costruire un training set e quindi sono esclusivamente la distribuzione e la struttura dei dati a determinare l'appartenenza o meno di un elemento a un determinato cluster.

In sostanza, si può affermare che gli algoritmi di Clusterings sono usati principalmente per le seguenti finalità:

1. Classificazione non supervisionata;

2. Scoperta di schemi concettuali usabili per il ragguppamento di entità;

3. Generazione di ipotesi tramite esplorazione dei dati;

4. Verifica di ipotesi o tentativo di determinare se raggruppamenti ipotizzati di oggetti sono realmente presenti nell'insieme dei dati;

5. Compressione dei dati.

1.1.2 Notazioni

Sia X l'insieme di dati da clusterizzare come:

$$X = \{\vec{x_i} | \vec{x_i} = (x_{i1}, ..., x_{iN})\}_{i=1..n}$$

con n la cardinalità dell'insieme X. Ogni elemento $\vec{x_i} \in X$ è un **punto** (o vettore di dimensione N) che rappresenta un oggetto. Ogni componente $x_{ij} \in X_j$, con j = 1,...,N è detta **attributo** (o **feature**) e N indica il numero di attributi che caratterizzano le entità che ci interessa raggruppare. Poichè ogni dato viene rappresentato come un vettore di attributi, l'insieme dei dati X può essere rappresentato come una opportuna matrice di dimensioni $n \times N$, dove ogni riga rappresenta un dato e ogni colonna un attributo.

1.1.3 Formulazione del Problema

Dato che l'obiettivo finale del clustering è quello di assegnare i punti di un insieme X di dati a un numero finito m di cluster, occorre organizzare i dati in m cluster, secondo opportuni criteri. In generale, richiederemo che i cluster prodotti non siano vuoti, che non si intersechino (non abbiano cioè elementi comuni) e che l'unione di tutti i sottoinsiemi ci fornisca l'insieme di dati originale (con alcune possibili eccezioni dovute ad elementi non classificabili, indicati in genere con il nome di 'outliers').
Formalmente, possiamo così definire l'obiettivo di un algoritmo di clustering. Assegnati:

- un insieme $X = \{\vec{x_1}, ..., \vec{x_n}\}$ di dati;
- il numero m di cluster cercati;
- una funzione obiettivo $F(\cdot)$ che valuti la qualità del clustering ottenuto.

si vuole determinare un'applicazione:

$$\gamma : X \rightarrow \{1, ..., m\}$$

che minimizzi la funzione obiettivo, nel rispetto di opportuni (eventuali) vincoli.
La funzione obiettivo che valuta la qualità del clustering è spesso definita in termini di similitudine o distanza tra i dati ed è di frequente indicata anche come *funzione di costo* o di *distorsione*. Ovviamente, la misura della similitudine è il parametro chiave di un algoritmo di clustering.

1.1.4 Applicazioni

Il Clustering è stato applicato un numero sterminato di campi, di cui qui elenchiamo solo i principali:

- Ingegneria (apprendimento automatico, pattern recognition, ingegneria meccanica, ingegneria elettronica). Le applicazioni tipiche del clustering in ingegneria spaziano dal riconoscimento biometrico al riconoscimento del linguaggio, all'analisi dei segnali radar, compressione dei dati e rimozione del rumore.

- Computer science. Le applicazioni in questo campo sono svariate, come intelligenza artificiale, estrazione dati su web, analisi spaziale dei database, recupero di informazioni, collezionamento di documenti testuali e segmentazioni di immagini.

- Scienze biologiche e mediche (genetica, biologia molecolare, microbiologia, paleontologia, psichiatria, analisi cliniche, filogenesi, patologia). In questi settori sono presenti alcune delle maggiori attuali applicazioni del clustering. Applicazioni importanti sono, tra le altre, definizioni

tassonomiche, identificazione delle funzioni di geni e proteine, diagnosi e trattamento di malattie.

- Astronomia e Scienze della Terra (geografia, geologia, rilevamento remoto). Il clustering può essere usato per classificare stelle e pianeti, investigare la formazione della terra, suddividere regioni e città e contribuire allo studio dei sistemi fluviali e montani.

- Scienze sociali (sociologia, psicologia, archeologia, antropologia, scienze dell'educazione). Interessanti applicazioni possono essere trovate nell'analisi delle caratteristiche del comportamento, nell'identificazione di relazioni all'interno di diverse culture, formazione ed evoluzione storica dei linguaggi, analisi dei social network, ritrovamenti archeologici, classificazione degli artefatti e studio della psicologia criminale.

- Economia (marketing, pianificazione di investimenti). Applicazioni come caratterizzazione della clientela, riconoscimento delle caratteristiche degli acquisti, raggruppamento di aziende e analisi dell'andamento delle scorte possono tutte beneficiare dell'uso della Cluster Analysis.

1.2 Misure di similarità

Benchè non ci sia un generale accordo e una precisa definizione sul termine 'cluster', tutte le definizioni presenti in letteratura descrivono il 'cluster' in termini di omogeneità interna ed separabilità esterna. Ovviamente, al fine di quantificare oggettivamente queste due grandezze, tutte le tecniche di clustering si basano sul concetto di distanza tra due elementi. Infatti, la bontà delle analisi ottenute dagli algoritmi di clustering dipende in maniera fondamentale dalla scelta della metrica, e quindi da come viene calcolata la

distanza: ad esempio, gli algoritmi di clustering agglomerativi raggruppano gli elementi sulla base della loro distanza reciproca, e quindi l'appartenenza o meno ad un insieme dipende da quanto l'elemento preso in esame è vicino all'insieme stesso, mentre gli algoritmi divisivi misurano l'omogeneità e la compattezza del singolo cluster per decidere, sempre tramite il concetto di distanza, quale cluster è conveniente suddividere in due sottocluster più piccoli.

1.2.1 Norme e metriche

Il modo più spontaneo e naturale di definire un criterio di similarità è quello di definire un'opportuna norma sullo spazio vettoriale di interesse.

Stabilire una **norma** su uno spazio vettoriale (reale o complesso X) significa assegnare una funzione $||.|| : X \to [0, +\infty)$ che verifichi le seguenti condizioni:

1. Funzione definita positiva:
   $||x|| \geq 0 \ \forall x \in X$ e $||x|| = 0 \Leftrightarrow x = 0$

2. Omogeneità:
   $||\lambda \cdot x|| = |\lambda| \cdot ||x|| \ \forall \lambda \in R$

3. Proprietà triangolare:
   $||x + y|| \leq ||x|| + ||y|| \ \forall x, y \in X$

Le norme più comunemente usate nello spazio euclideo R^N (di dimensione N) sono:

1. Norma 1 - L_1:
   $||(x_1, ..., x_N)||_1 := |x_1| + ... + |x_N|$

2. Norma euclidea - L_2:
   $||(x_1, ..., x_N)|| := \sqrt{x_1^2 + ... + x_N^2}$

3. Norma p - L_p:
   $||(x_1, ..., x_N)||_p := \sqrt[p]{|x_1|^p + ... + |x_N|^p}$

4. Norma infinito - L_∞:
   $||(x_1, ..., x_N)||_\infty := max\{|x_1|, ..., |x_N|\}$

Una volta definita una norma, si può subito ottenere una **metrica** (o **distanza**) considerando la metrica indotta dalla norma:

$$d(x, y) := ||x - y|| \ \forall x, y \in X$$

Si vede subito che tale scelta rispetta tutte le proprietà di una distanza e a tal fine ricordiamo che uno **spazio metrico** è una struttura matematica costituita da una coppia (X,d) di elementi, dove X è un insieme e d una funzione distanza (detta anche metrica), che associa a due punti x e y di X un numero reale non negativo $d(x,y)$ in modo che le seguenti proprietà valgano $\forall\ x, y, z \in X$:

1. Positività:
   $d(x, y) \geq 0$

2. Identità:
   $d(x, y) = 0 \Leftrightarrow x = y$

3. Simmetria:
   $d(x, y) = d(y, x)$

4. Disuguaglianza triangolare:
   $d(x, y) \leq d(x, z) + d(z, y) \ \forall x, y, z \in X$

1.2.2 Definizione di Centroide

La prima applicazione del concetto di distanza si ha negli algoritmi agglomerativi dove, dato un cluster (ottenuto in qualche maniera) è necessario definire un suo 'rappresentante', cioè un elemento che, dipendendo fortemente dalla sua composizione, ne descriva oggettivamente omogeneità e compattezza. A tal fine, stabilita la metrica da usare, uno dei modi più comuni è quello di definire il '**Centroide**' (o '**Medioide**'), elemento che, pur rappresentando sinteticamente il cluster, può non appartenere ad esso.
In maniera più formale, dato un insieme A di n vettori $x_1, x_2, ..., x_n \in R^N$, nello spazio R^N, dotato di norma $||.||_p$ $(p \geq 1)$, con '**Centroide**' si intende il vettore C_A tale che:

$$C_A = arg\,\min_{c \in R^N} \sum_{i \in A} ||x_i - c||^p$$

Si rimanda alle **Appendici A** e **B** per le proprietà del Centroide e della Funzione obiettivo su di esso definita.
Rappresentando il cluster in maniera sintetica, in svariati algoritmi il centroide è usato come base per definire distanze più semplici da calcolare o più sofisticate, per particolari finalità applicative.

1.2.3 Metriche per la valutazione della distanza tra i cluster

Una volta stabilita la metrica, dati due cluster si possono scegliere vari modi con cui valutare la loro distanza. Le più usate negli algoritmi agglomerativi sono:

1. **Single-link proximity.** Calcola la distanza tra i due cluster come la distanza minima tra elementi appartenenti a cluster diversi (cioè la distanza di due cluster coincide con quella dei loro elementi più vicini):

$$d(C_i, C_j) = min_{x \in C_i, y \in C_j} d(x, y)$$

2. **Average-link proximity.** Questa funzione calcola la distanza tra i due cluster come la media delle distanze tra i singoli elementi:

$$d(C_i, C_j) = \frac{1}{|C_i| \cdot |C_j|} \sum_{x \in C_i, y \in C_j} d(x, y)$$

3. **Complete-link proximity.** Questa funzione calcola la distanza tra i due cluster come la distanza massima tra elementi appartenenti ai due cluster (cioè la distanza di due cluster coincide con quella dei loro elementi più lontani):

$$d(C_i, C_j) = max_{x \in C_i, y \in C_j} d(x, y)$$

4. **Distanza tra centroidi.** Definito un baricentro, la distanza tra i due cluster coincide con la distanza calcolata tra i centroidi (medioidi o baricentri):

$$d(C_i, C_j) = d(c_i^*, c_j^*)$$

Nel clustering divisivo, invece, è necessario individuare il cluster da suddividere in due sottogruppi e quindi sono necessarie funzioni che misurino la compattezza del cluster, la densità o la sparsità dei punti assegnati ad un cluster. Le funzioni normalmente utilizzate nel caso divisivo sono:

(a) **Average internal similarity.** Questa funzione valuta la similarità media tra gli elementi interni ad un cluster: più sono tra loro dissimili (valori bassi di similarità), più il cluster è da suddividere in sottogruppi:

$$d(C_i) = \frac{1}{|C_i| \cdot (|C_i| - 1)} \sum_{x,y \in C_i, x \neq y} d(x, y)$$

(b) **Maximum internal distance.** Questa funzione valuta la distanza massima tra due punti interni ad un cluster. Tale valore è noto anche come 'diametro del cluster': più tale valore è basso, più il cluster è compatto:

$$d(C_i) = max_{x,y \in C_i} d(x, y)$$

È importante sottolineare che le tecniche di Clustering non predittive non consentono un giudizio assoluto ed oggettivo sulla loro efficacia: in altri termini, i cluster in cui raggruppare i campioni sono determinati sulla base di misure di similarità **spesso soggettivamente scelte**, e questa scelta è proprio basata sull'abilità dell'algoritmo di creare cluster 'interessanti' cioè simultaneamente omogenei e ben separati. È quindi evidente come sia cruciale la scelta delle misure di similarità adottate, quando si nota che lo stesso algoritmo, elaborando il medesimo insieme di dati, può generare cluster completamente diversi usando un differente criterio di similarità. In aggiunta, non esiste un modo universalmente valido per determinare in generale il miglior criterio per una specifica applicazione: infatti, ogni criterio ha il suo uso appropriato a seconda della particolare occasione, sebbene alcuni di essi risultino essere applicabili in un maggior numero di casi rispetto ad altri.

1.3 Metodi di Clustering

I metodi di Clustering possono essere classificati ad alto livello in pochi gruppi principali, secondo tassonomie che si differenziano in relazione ai criteri utilizzati per caratterizzare i diversi algoritmi.
Di seguito, vengono riportate alcune delle tassonomie comunemente adottate in letteratura.

1.3.1 Tassonomie dei metodi di clustering

A seconda della tecnica scelta, della modalità di selezione dei cluster, della possibilità che un elemento possa o meno essere simultaneamente assegnato a più cluster, e in relazione alla tipologia della procedura utilizzata per dividere lo spazio, gli algoritmi di clustering si possono classificare come segue:

(a) **Algoritmi agglomerativi e divisivi**: questi algoritmi assumono inizialmente che ogni cluster (foglia) contenga un singolo punto; ad ogni passo, poi, vengono fusi i cluster più vicini fino ad ottenere un numero prefissato di cluster, oppure fino a che la distanza minima tra i cluster non superi un certo valore soddisfacente la condizione di terminazione. Questi algoritmi necessitano di misure per valutare la similarità tra cluster onde poter scegliere, ad ogni passo, la coppia di cluster da fondere.

Gli algoritmi divisivi, invece, partono considerando lo spazio organizzato in un singolo grande cluster contenente tutti i punti. Suddividendolo poi ricorsivamente in due ad ogni passo, in base ad una opportuna misura, viene selezionato un cluster che viene

suddiviso in due cluster più piccoli. Normalmente viene fissato un numero minimo di punti sotto il quale il cluster non viene ulteriormente suddiviso (nel caso estremo questo valore è 1), oppure più in generale si adottano misure di 'dispersione' o 'compattezza' del cluster, per decidere se procedere ad ulteriori suddivisioni o meno. Questi tipi di algoritmi necessitano di una funzione per scegliere il cluster da suddividere, ed il criterio che guida la divisione è sempre quello di cercare di ottenere cluster con elementi il più possibile omogenei. L'algoritmo procede fino a che non ha raggiunto un numero prefissato di cluster. Una rappresentazione grafica dei processi di clustering agglomerativi o divisivi è fornita dal Dendrogramma.

(b) **Algoritmi di Clustering esclusivo o non-esclusivo**: negli algoritmi esclusivi ogni elemento può essere assegnato ad uno ed ad un solo gruppo. I cluster risultanti, quindi, non possono avere elementi in comune. Questo approccio è detto anche Hard Clustering.

Negli algoritmi non esclusivi, invece, un elemento può appartenere a più cluster con gradi di appartenenza diversi. Questo approccio è noto anche con il nome di Soft Clustering o Fuzzy Clustering. Un algoritmo non-esclusivo può ovviamente essere convertito in uno esclusivo tramite assegnamento di ogni elemento al cluster che lo presenti con il maggiore grado di appartenenza. Ulteriori esempi sono rappresentati da algoritmi probabilistici e possibilistici.

(c) **Algoritmi monotetici o politetici**: questo aspetto è relativo all'uso sequenziale o simultaneo delle feature durante il calcolo delle distanze tra i pattern. La maggioranza degli algoritmi sono po-

litetici e usano tutte le feature. Un semplice algoritmo monotetico riportato in [Anderberg 1973] considera le feature sequenzialmente per partizionare i pattern assegnati: il suo maggior incoveniente è che genera 2^N cluster, se N è la dimensionalità dei pattern. Per valori di N comunemente usati, tali algoritmi generano un numero ingestibile di cluster, generalmente di dimensioni troppo ridotte per essere informativi.

(d) **Algoritmi deterministici o stocastici**: gli algoritmi deterministici durante la loro esecuzione non fanno uso di quantità aleatorie e questo implica che, se attivati sullo stesso ingresso, diano sempre lo stesso risultato, mentre quelli stocastici sugli stessi ingressi possono dare risultati differenti. Le tecniche stocastiche sono in genere adoperate quando lo spazio delle potenziali soluzioni è troppo ampio per una ricerca deterministica esaustiva: ad esempio, queste tecniche sono particolarmente rilevanti nel progetto di algoritmi partizionali al fine di ottimizzare una funzione di costo quadratica. Questa ottimizzazione infatti può essere ottenuta tramite tecniche tradizionali oppure attraverso una ricerca random nell'intero spazio degli stati.

(e) **Algoritmi incrementali o non-incrementali**: questo aspetto è fondamentale quando l'insieme da clusterizzare è molto grande oppure se esistono vincoli sul tempo di esecuzione o sullo spazio di memoria utilizzabile. Occorre sottolineare che solo recentemente (sollecitati dall'avvento del data mining e delle applicazioni in campo bioinformatico) si sono sviluppati algoritmi efficienti su insiemi di dati molto ampi o di elevata dimensione: ciò ha imposto lo sviluppo di tecniche di progetto finalizzate alla realizzazione di

algoritmi che minimizzassero il numero di accessi ai dati, riducendo il numero di pattern esaminati durante l'esecuzione o usando strutture dati ottimizzate di minor dimensioni. Un'appropriata osservazione in [Jain e Dubes 1988] è che la specificazione di un algoritmo di clustering deve di solito permettere una considerevole flessibilità implementativa.

Naturalmente, queste sono solo le principali caratteristiche in base alle quali gli algoritmi di Clustering possono essere classificati e nel seguito vedremo alcune delle principali tipologie di algoritmi proposte in letteratura.

1.3.2 Algoritmi di Clustering Gerarchici

Algoritmi di questo tipo sono usati quando l'insieme delle entità di partenza non possiede una singola e naturale partizione in cluster ben separati.

Questi metodi organizzano i dati in una struttura gerarchica basata su di una matrice di prossimità. Il risultato di un algoritmo di Clustering Gerarchico è in genere illustrato tramite un albero binario detto anche '**dendrogramma**', la cui radice rappresenta l'intero insieme dei dati di partenza e ogni nodo foglia è un singolo punto dell'insieme dei dati. I nodi intermedi costituiscono i cluster (generati ad ogni iterazione) e forniscono una rappresentazione della misura con la quale gli oggetti sono prossimi agli altri, mentre l'altezza del dendrogramma esprime la distanza tra ogni coppia di punti dei dati, tra ogni coppia di cluster oppure tra un punto dei dati e un cluster. Il risultato terminale dell'algoritmo può essere ottenuto tagliando il dendrogramma a diffe-

renti livelli, ottenendo una soluzione composta da differenti cluster. In genere, tali algoritmi sono usati, con buoni risultati, quando nei dati è naturalmente presente una struttura gerarchica, come, ad esempio, nei dati biologici derivanti da analisi genica su diverse specie di organismi viventi a vari livelli della scala evolutiva, oppure in altre applicazioni di medicina, biologia e archeologia.

A seconda della tecnica adoperata per costruire il dendrogramma, gli algoritmi gerarchici a loro volta si dividono in due sottoclassi:

(a) Gerarchici agglomerativi (approccio Bottom-Up);

(b) Gerarchici divisivi (approccio Top-Down).

I primi assegnano ogni elemento in ingresso ad un singolo cluster e, iterativamente, cercano di determinare i due cluster più simili per poterli fondere in un unico cluster. La procedura termina quando si giunge ad un unico cluster contenente tutti i dati. La partizione migliore si ottiene applicando un opportuno criterio di validazione, identificando in pratica a quale livello dell'albero si ha la migliore clusterizzazione.
Gli algoritmi gerarchici divisivi procedono in maniera inversa rispetto agli agglomerativi, partendo da un unico cluster contenente tutti gli elementi in ingresso e continuando a suddividerlo attraverso ripetute bipartizioni, fino a trovare, al limite, cluster con un solo elemento. È importante notare che, applicando tale metodo, non sempre si riesce ad ottenere l'ottimo: usando un opportuno criterio di validazione, si cerca di identificare a quale livello dell'albero si ha la partizione ottimale. In tale direzione, il primo algoritmo separativo per il criterio di somma dei quadrati (metrica L_2) fu proposto in [Edwards e Cavalli-Sforza 1965]:

esso risolveva il problema tramite esplicita enumerazione di tutte le possibili bipartizioni dell'insieme delle entità considerate.

Uno dei principali vantaggi di questi metodi è la loro capacità di restituire un numero arbitrario di cluster, che meglio descrivano la naturale struttura dei dati in ingresso. Il loro principale svantaggio, oltre la loro sensibilità a valori fuori scala, è la loro complessità computazionale e questa caratteristica limita notevolmente la loro applicabilità a dati di grandi dimensioni. Per queste ragioni, gli algoritmi agglomerativi sono maggiormente usati in pratica e, benchè sia evidente che gli algoritmi gerarchici separativi possono essere usati solo per istanze di piccola dimensione, essi sono stati molto studiati al fine di migliorare il metodo di completa enumerazione per bipartizioni, e questo ha condotto sostanzialmente a due filoni di ricerca. In primo luogo, sono state esplorate le proprietà geometriche delle bipartizioni ottimali: infatti Edwards e Cavalli-Sforza [Edwards e Cavalli-Sforza 1965] notarono che la somma dei quadrati per un singolo cluster (metrica euclidea L_2) è uguale alla somma delle distanze al quadrato tra tutte le coppie delle entità del cluster, diviso il numero delle entità stesse. Scott e Symons [Scott e Symons 1971] osservarono che, dai risultati di Fisher [Fisher 1958] e Grower [Grower 1967], si poteva dedurre che le bipartizioni per la somma dei quadrati fossero contigue, nel senso che ogni punto dell'involucro convesso dell'insieme di punti di un cluster apparteneva a quel cluster. Inoltre, Harding [Harding 1967], notò che i punti di una tale bipartizione potevano essere separati tramite un iperpiano, e Grower [Grower 1967] propose un algoritmo euristico basato su questi risultati geometrici. In secondo luogo, è stato analizzato l'uso e i vantaggi della programmazione intera: Rao [Rao 1971] formulò il

problema di bipartizione di Edwards e Cavalli-Sforza in termini di un problema frazionario non lineare in variabili 0-1. Occorre sottolineare che, se all'epoca non esistevano algoritmi efficienti, recentemente i progressi in programmazione quadratica e non lineare hanno reso la formulazione di Rao più attraente.

Un'ulteriore divisione può essere fatta a seconda del modo con il quale gli algoritmi gerarchici caratterizzano la similarità tra coppie di cluster:

(a) Algoritmi single-link [Jain e Dubes 1988, Sneath e Sokal 1973]: utilizzano come distanza tra due cluster il minimo della distanza tra tutte le coppie di punti appartenenti ai cluster in esame. Benchè più versatili degli algoritmi complete-link, essi soffrono del cosiddetto 'effetto catena' [Nagy 1968], cioè tendono a produrre cluster sottili ed allungati.

(b) Algoritmi complete-link: [King 1967]: calcolano come distanza tra due cluster il massimo della distanza tra tutte le coppie di punti appartenenti ai cluster in esame, producendo cluster strettamente limitati e compatti: questa caratteristica fornisce, in molte applicazioni, gerarchie di cluster maggiormente utilizzabili [Jain e Dubes 1988].

1.3.3 Algoritmi partitivi hard

Gli algoritmi Partitivi mirano ad ottenere una singola partizione invece di un insieme di strutture di clustering (come gli algoritmi gerarchici) e quindi sono vantaggiosi in quelle applicazioni in cui le grandi dimensioni dell'insieme dei dati rende proibitiva la costruzione dei dendrogrammi.

In pratica, l'insieme delle entità viene diviso in un numero m (prefissato) di cluster che sostanzialmente costituiscono una sua partizione: nessuno dei m cluster deve risultare vuoto, la loro unione deve restituire l'insieme di partenza e, soprattutto, ogni elemento deve essere assegnato ad uno e ad un solo cluster. Per realizzare la partizione ottima questi algoritmi definiscono sul cluster una misura di similarità, a sua volta usata per definire la funzione obiettivo da minimizzare. In genere, le procedure adottate sono di tipo iterativo: a seconda della funzione e della procedura adottata si possono avere vari tipi di algoritmi e l'ottimo viene cercato tra i minimi locali, i quali costituiscono le soluzioni sub-ottime.
In maniera più formale, il problema può allora essere formulato come segue:

Problema: m-Clustering
Istanza: un insieme A di vettori $x_1, x_2, ..., x_n \in \mathrm{R}^N$, un intero m.
Soluzione: $arg\min_{\langle B_1,...B_m\rangle} F_A(B_1, B_2, ..., B_m)$
dove $(B_1, B_2, ..., B_m)$ è una partizione di $\{1, 2, ..., n\}$ e $F_A(\cdot)$ è un'opportuna funzione obiettivo da ottimizzare. La funzione obiettivo viene comunemente definita in base alla metrica L_p $(p \geq 1)$ usata, e la metrica L_2 (euclidea) è stata storicamente la prima ad essere adottata, sostanzialmente per la sua semplicità.
Sono ovviamente possibili varie scelte per definire la funzione obiettivo per un determinato problema ma, generalmente, la funzione obiettivo è formata da una componente che esprime quanto è compatto e omogeneo il singolo cluster e poi da un'altra componente, che quantifica globalmente la performance dell'intera partizione. La realizzazione

della funzione obiettivo prevede la definizione di un centroide rappresentativo del singolo cluster, calcolato con un'opportuna metrica e la misurazione della compattezza del cluster tramite la valutazione degli scarti (pesati anch'essi tramite una metrica opportuna) degli elementi del cluster dal loro centroide rappresentante. La funzione $F_A(\cdot)$ valuta infine la performance della partizione complessiva sommando aritmeticamente le compattezze di tutti i singoli cluster.

I primi studi sul Clustering prendevano in considerazione solo il caso in cui si adoperava la medesima metrica (in particolare L_2) sia per definire il centroide nel cluster che per quantificare la misura di dissimilarità tra il centroide e gli elementi del cluster, ma recentemente in Novick [Novick 2009] sono stati presi in considerazione problemi con norme miste $L_p - L_q$, la prima utilizzata per il centroide, la seconda per misurare la dissimilarità.

L'algoritmo più famoso appartenente a questa famiglia è il K-means, proposto da MacQueen [MacQueen 1967]: inizialmente viene generata una partizione casuale e gli elementi vengono poi riassegnati ai cluster finchè il criterio di convergenza (usante metriche L_2) non è soddisfatto: in genere, i cluster sono prodotti tramite l'ottimizzazione di una funzione di costo scelta localmente (su un sottoinsieme dei pattern) o globalmente (su tutti i pattern). Tale algoritmo è stato ampiamente applicato in molti contesti perchè di facile implementazione e la sua complessità è $O(m \cdot n)$, con m numero di cluster e n numero di elementi: diverse varianti di esso sono riportate in letteratura [Anderberg 1973]. Altri algoritmi abbastanza noti, appartenenti a questa classe sono ISODATA, proposto da Ball e Hall [Ball e Hall 1965] e il Dynamic Clustering Algorithm [Diday 1973, Symon 1977].

A dispetto della loro semplicità, questi algoritmi presentano alcuni inconvenienti: in primo luogo, un problema che si pone nel loro uso è la scelta opportuna del numero m di cluster desiderato (fissato a priori) [Dubes 1987] ed inoltre, considerato che n elementi possono essere ripartiti in m cluster in

$$\frac{1}{m!}\sum_{i=1}^{m}(-1)^{m-i}C_m^i i^n$$

modi distinti, dove $C_m^i = \binom{m}{i}$, vi è l'impossibilità di cercare la soluzione ottimale per enumerazione, dato che la ricerca cominatoria dell'ottimo è computazionalmente proibitiva (anche per dati in ingresso di dimensione relativamente piccola): questa circostanza obbliga all'uso di euristiche per la ricerca di soluzioni approssimate. Poichè la convergenza verso l'ottimo globale non può essere ovviamente garantita, questi metodi sono molto sensibili alle condizioni iniziali, variando le quali l'algoritmo può convergere a un diverso ottimo locale. Pertanto, in pratica, l'algoritmo viene utilizzato facendolo girare con diverse condizioni iniziali e selezionando alla fine la migliore tra le risposte ottenute. In aggiunta, questi metodi tendono a generare cluster di forma sferica e sono abbastanza sensibili al rumore, nel senso che si lasciano facilmente influenzare da quei punti dell'insieme di partenza che non appartengono naturalmente a nessuno dei cluster.

Per ovviare ad alcuni di questi problemi, possibili varianti sono costituiti dagli algoritmi 'Density based', in cui l'idea base è quella di considerare come cluster potenziali i sottoinsiemi dell'insieme di partenza 'densamente popolati': questi sottoinsiemi potranno avere forma arbitraria e potranno ragionevolmente essere considerati separati da ogni altro presente nell'insieme di partenza. Altre soluzioni possibili sono

gli algoritmi basati sulla Teoria dei Grafi, che possono essere vantaggiosamente usati se i cluster cercati possono presentare particolari tipi di strutture. Ulteriori possibili varianti degli algoritmi m-Clustering partizionali cercano di assegnare un oggetto a più cluster simultaneamente, con un diverso grado di appartenenza: questi algoritmi sono noti come Fuzzy m-Clustering.

1.3.4 Algoritmi Mixture-resolving e Mode-Seeking

Il tentativo di usare algoritmi Mixture-resolving per il problema del Clustering è stato sviluppato in molti modi: l'ipotesi di base è che gli oggetti da clusterizzare provengano da una di parecchie distribuzioni e l'obiettivo è individuare i parametri di ognuna e, magari, anche il loro numero. La maggior parte dei lavori in tal senso assume che le componenti individuali delle densità miste siano Gaussiane e quindi il procedimento tenta di stimare i parametri di queste distribuzioni. Tradizionalmente, l'approccio a questi problemi consiste nell'ottenere (iterativamente) la stima maggiormente probabile del vettore dei parametri delle distribuzione componenti [Jain e Dubes 1988]: più recentemente, sono stati applicati al problema di stima parametrica gli algoritmi di Expectation Maximization (EM) [Dempster 1977, Mitchell 1997], non escludendo d'altronde l'uso di tecniche non parametriche [Jain e Dubes 1988].

1.3.5 Nearest Neighbor Clustering

Poichè il concetto di 'prossimità' gioca, ovviamente, un ruolo fondamentale nella nozione di cluster, le distanze tra gli elementi più vicini possono servire come base di una tecnica di cluster: una procedura iterativa fu proposta da Lu e Fu [Lu e Fu 1978]; essa assegna ogni elemento non classificato al suo più vicino elemento etichettato, assicurandosi che tale distanza sia minore di una fissata soglia. Il processo continua fino ad avere etichettato tutti gli elementi o finchè non necessitano più etichette addizionali.

1.3.6 Fuzzy Clustering

L'approccio tradizionale al clustering genera partizioni, cioè ogni elemento appartiene ad uno e un solo cluster. Pertanto, in un algoritmo di hard clustering i cluster ottenuti risultano essere disgiunti. Gli algoritmi di Fuzzy clustering estendono questa nozione attribuendo ogni elemento ad ogni cluster tramite l'uso di una opportuna funzione di appartenenza [Zadeh 1965]. Naturalmente un hard clustering può essere ottenuto da un fuzzy clustering quantizzando a gradini la funzione di appartenenza. La teoria degli insiemi fuzzy fu applicata al clustring per la prima volta da Ruspini [Ruspini 1969].

1.3.7 Reti Neurali Artificiali per il Clustering

Le Reti Neurali Artificiali (ANNs) [Hertz et al. 1991] si sono sviluppate dallo studio delle reti neurali biologiche, e sono state diffusamente usate negli ultimi trent'anni sia per la classificazione che per il clustering

[Sethi e Jain 1991, Jain e Mao 1994]. Le caratteristiche che rendono le ANNs importanti in clustering sono le seguenti:

(a) Analizzano vettori numerici e quindi richiedono pattern rappresentati usando solo feature quantitative.

(b) Sono ad archiettura intrinsecamente parallela e distribuita.

(c) Possono apprendere modificando adattivamente i pesi delle loro interconnessioni [Jain e Mao 1996, Oja 1982].

Le Reti Neurali competitive [Jain e Mao 1996] sono spesso usate per clusterizzare dati: pattern similari vengono raggruppati tramite la rete e rappresentati come una singola unità (neurone) e tale raggruppamento è automaticamente basato sulle correlazioni dei dati, ed esempi ben noti sono le mappe di Kohonen e le mappe autorganizzanti (SOM) [Kohonen 1989]. Generalmente le architetture di questi ANNs sono semplici: sono a strato singolo, i pattern sono presentati in input e sono associati con i nodi di output. I pesi tra i nodi di input e quelli di output sono cambiati iterativamente (questo è chiamato 'apprendimento') finchè è soddisfatta una condizione di terminazione.
Uno dei problemi presenti in questo tipo di approccio è che esistono particolari input che possono accendere differenti unità di output a differenti iterazioni: questa circostanza porta a definire la stabilità del sistema di apprendimento. Il sistema è detto **stabile** se nei dati di addestramento non ci sono pattern che cambiano la loro categoria dopo un finito numero di cicli di apprendimento. Questo problema è strettamente connesso con il problema della **plasticità**, la quale consiste sostanzialmente nell'abilità dell'algoritmo di adattarsi a nuovi dati. A causa della stabilità, il tasso di apprendimento tenderà a zero all'au-

mentare delle iterazioni e questo naturalmente influisce sulla plasticità dell'algoritmo. Un'altro problema che questi algoritmi possono presentare è quello di essere ordine-dipendenti, nel senso che differenti partizioni sono ottenute variando l'ordinamento in cui i medesimi dati sono presentati in input alla rete. Inoltre, sia la misura che il numero dei cluster generati dipendono dal valore scelto per la soglia di vigilanza, il quale è usato per decidere se un pattern sia assegnato a un cluster già esistente o debba costituire un nuovo cluster. In ogni caso, le reti neurali a singolo strato riescono a distinguere solo cluster ipersferici, mentre per rilevare cluster iperellissoidali è necessaria un'architettura a due strati con la distanza regolarizzata di Mahalanobis (Mao e Jain [Mao e Jain 1995]).

1.3.8 Algoritmi Evolutivi per il Clustering

Direttamente ispirati dalla biologia, approcci evolutivi usano operatori evolutivi e una popolazione di soluzioni per ottenere la partizione ottimale globale dei dati. Le soluzioni candidate per il problema del Clustering sono codificate come cromosomi e sottoposti a operatori evolutivi come selezione, ricombinazione e mutazione, mentre una funzione di idoneità valuta la probabilità di sopravvivenza di un cromosoma nella successiva generazione. Le tecniche evolutive meglio conosciute sono gli Algoritmi Genetici (GAs) [Holland 1975, Goldberg 1989], le strategie evolutive (ESs) [Schwefel 1981] e i programmi evoluzionisti (EP) [Fogel et al. 1965] e tutte sono state usate per risolvere il problema del Clustering concepito come la minimizzazione di un criterio di errore quadratico.

Il maggior problema degli Algoritmi Genetici (e, in generale, di tutti gli algoritmi evolutivi) è costituito dalla loro sensibilità alla selezione di vari parametri quali la misura della popolazione, le probabilità delle mutazioni, eccetera: in Grefenstette [Grefenstette 1986] è suggerita una linea guida per la selezione di questi parametri di controllo, peraltro non completamente soddisfacente per specifici problemi, per i quali si ottengono risultati migliori incorporando negli algoritmi genetici (ibridi) euristiche specifiche del problema in esame (Jones e Beltramo [Jones e Beltramo 1991]).

1.3.9 Algoritmi Search-Based

Gli algoritmi per ottenere il valore ottimo della funzione obiettivo basati su tecniche di ricerca si dividono sostanzialmente in due categorie:

(a) **Algoritmi deterministici**, che garantiscono l'ottenimento della partizione ottimale attraverso una enumerazione esaustiva delle soluzioni e tipicamente in questi approcci la funzione obiettivo descresce (per la ricerca del minimo) in maniera monotona (Greedy). Tra gli altri, ricordiamo in questa categoria le tecniche Branch-and-Bound, adottate per il clustering in Koontz [Koontz et al. 1975] e Cheng [Cheng 1995], e quelle di Annealing deterministico in Rose [Rose et al. 1993]. Tali approcci non sono frequentemente usati proprio a causa del loro eccessivo costo computazionale (anche su istanze di dimensioni relativamente piccole).

(b) **Algoritmi stocastici**, i quali invece pur non garantendo il raggiungimento dell'ottimo, generano una partizione sub-ottima in maniera ragionevolmente veloce garantendo la convergenza asin-

totica alla partizione ottimale. In genere, la loro filosofia è quella di permettere perturbazioni delle soluzioni (con probabilità non nulla) anche in direzioni non localmente ottimali. Tra queste tecniche ricordiamo l'Annealing simulato, applicato al clustering in Klein e Dubes [Klein e Dubes 1989]: esse cercano di evitare di bloccarsi in soluzioni che corrispondono solo a minimi locali della funzione obiettivo. Questo è ottenuto accettando, con probabilità non nulla, che la nuova soluzione alla successiva iterazione sia di qualità peggiore e tale probabilità è governata da un parametro critico chiamato, in analogia con il fenomeno fisico, 'temperatura' (Selim e Al-Sultan [Selim e Al-Sultan 1991]). Ovviamente, l'ottenimento della partizione ottima globale è garantita solo in senso statistico.

Riepilogando, ogni algoritmo di Cluster Analysis (al di là delle specifiche implementazioni) è costituito essenzialmente da quattro componenti:

(a) **Selezione delle caratteristiche (feature) o estrazione:** le entità vengono esaminate per identificare quelle feature che possono essere significative per l'analisi e/o si cerca sottoporle a qualche traformazione (estrazione) al fine di generarne di nuove e più utilizzabili. Il rischio di quest'ultima operazione, rispetto alla selezione, è quello di creare feature che non siano fisicamente interpretabili, mentre un'opportuna selezione ed un'efficace elaborazione posso ridurre notevolmente sia la quantità dei dati memorizzati richiesti che il tempo di elaborazione, oltre a semplificare il progetto dell'algoritmo e, soprattutto, a migliorare la comprensione dei dati. In

sostanza, le feature ideali dovrebbero essere usabili in distinguibili pattern all'interno di differenti cluster, possibilmente immuni al rumore e facili da ottenere e interpretare. Come facilmente comprensibile, la feature selection è più spesso usata nel contesto della classificazione supervisionata, dove l'insieme delle caratteristiche salienti è noto a priori.

(b) **Progetto dell'algoritmo di Clustering (selezione):** sostanzialmente, questo stadio consiste nel determinare un'appropriata misura di prossimità, nel costruire una funzione di costo e nella sua minimizzazione mediante un opportuno algoritmo di clustering. Tutti gli algoritmi di Cluster Analysis devono avere, esplicitamente o implicitamente, una misura di vicinanza per stabilire il grado di rassomiglianza di due entità e/o cluster: a questo punto, stabilita la misura, il problema di clustering può essere modellato come un problema di ottimizzazione con un'opportuna funzione costo, la quale condiziona fortemente (ed inevitabilmente) la tipologia dei cluster selezionati.

(c) **Validazione dei Cluster:** ogni algoritmo di clustering, agendo su un insieme di dati, produce una clusterizzazione indipendentemente dal fatto che ci sia o meno una struttura in essi, ed il medesimo algoritmo, con una differente scelta dei parametri, in generale selezionerà altri cluster. Di qui la necessità di valutare se esistono cluster nei dati, ed in caso di affermazione positiva, valutare il loro numero e la loro affidabilità: tramite l'uso di opportuni indici (interni, esterni e relativi) è possibile quantificare la validità delle soluzioni ottenute.

(d) **Interpretazione dei risultati:** l'ultimo stadio dell'algoritmo è

fornire agli utenti una chiara comprensione dei dati e la soluzione effettiva del problema affrontato, suggerendo fondate ipotesi da verificare con ulteriori analisi ed esperimenti.

Capitolo 2

Clustering in R^1: Localizzazione del Centroide

2.1 Introduzione

Un sottoproblema preliminare al Clustering (e al Clustering vincolato) è quello della determinazione del p-Centroide di n punti, rispetto a $||.||_p$. La soluzione esatta di tale problema è solo apparentemente semplice, come mostriamo in questo capitolo cercando di classificare, dal punto di vista della complessità computazionale, il problema della localizzazione del p-Centroide: dati n numeri razionali $x_1, x_2, ..., x_n$ ed un numero razionale r, decidere se $C_p < r$, dove C_p è il p-Centroide di $x_1, x_2, ..., x_n$.

Per rendere il capitolo autocontenuto, inizialmente vengono richiamate alcune nozioni di complessità computazionale. In particolare, saranno introdotte le classi **P** (problemi risolubili in tempo polinomiale con algoritmi deterministici), **NP** (problemi risolubili in tempo polinomiale con

algoritmi non deterministici), **PP** problemi risolubili in tempo polinomiale con algoritmi probabilistici, senza limiti all'errore) e **CH** (classe introdotta da Wagner nel 1986 [Wagner 1986], per cogliere concetti di complessità legati a problemi di conteggio). Viene poi introdotta la nozione di problema difficile o completo per una classe, attraverso il concetto di riduzione tra problemi.

La non banalità del problema della localizzazione del p-Centroide è legata al fatto che ad esso, per $p = \frac{3}{2}$, si può ridurre polinomialmente il problema SQRT-Sum. Il problema del SQRT-Sum richiede di decidere, dati interi $n_1, n_2, ..., n_n$ e un intero t, se $\sum_i^n \sqrt{n_i} > t$. La determinazione della complessità computazionale di tale problema è stata posta da GaJo nel 1976 [Garey Graham Johnson 1976], in relazione allo studio del Problema del Commesso Viaggiatore Euclideo. A dispetto di molti sforzi, il miglior risultato per SQRT-Sum lo pone in **CH** [ABKM 2006].

I risultati ottenuti in questo capitolo sono:

(a) Per ogni p razionale, il Problema della Localizzazione del p-Centroide $\in$ **CH**;

(b) Se p intero, il Problema della Localizzazione del p-Centroide $\in$ **P**.

Poichè questi risultati richiedono metodi di compressione aritmetica, saranno richamate alcune nozioni e risultati relativi agli Straight-line Program.

2.2 Richiami di Complessità computazionale

Al contrario della Teoria della Calcolabilità, il cui scopo è stabilire se, dato un certo problema, esiste almeno un algoritmo risolutivo (cioè, informalmente, una procedura uniforme che, data una qualunque istanza, risolva il problema in un numero finito di passi), la Teoria della Complessità computazionale (cfr. [Bernasconi Codenotti 1998]) si pone il problema, accertata la risolubiltà algoritmica di un certo problema, se esso possa essere risolto da algoritmi efficienti o stabilire se essi possano essere risolti solo a patto di utilizzare una quantità irragionevole di risorse.
In maniera meno informale, le valutazioni di complessità dei problemi dipendono sostanzialmente da tre elementi:

(a) Il modello di calcolo adottato;

(b) La risorsa il cui uso si intende analizzare (in genere il tempo di calcolo richiesto o lo spazio di memoria utilizzato);

(c) Il criterio con cui misurare il costo associato all'uso della risorsa scelta.

I modelli di calcolo che vengono presi in considerazione devono essere abbastanza semplici per poter essere facilmente trattati matematicamente ma simultaneamente abbastanza potenti ed aderenti alla realtà, in modo che i risultati ottenuti sul modello scelto siano indipendenti da esso ed utili anche per macchine reali: la Macchina di Turing (MdT), pur essendo in apparenza molto diversa da un calcolatore reale, ne modella molto bene le caratteristiche essenziali e, in virtù della sua

semplicità matematica, viene ad essere il modello di riferimento nella Teoria della Complessità computazionale.

La prestazione di un algoritmo è definita in termini di quantità di risorsa richiesta (il tempo di calcolo e/o lo spazio di memoria) per risolvere il problema su un dato ingresso, utilizzando spesso la misura del 'caso peggiore'. I motivi di tale scelta sono evidenti: per ogni intero positivo n, si considera la massima quantità di risorsa necessaria all'algoritmo per risolvere il problema su ingressi di 'dimensione' n.
A differenza della Teoria della Calcolabilità, i risultati ottenuti in Teoria della Complessità sono notevolmente più deboli perchè, nel momento in cui si pongono dei limiti sulle risorse da utilizzare, è molto difficile ottenere risultati di separazione (cioè limitazioni inferiori di complessità) non banali. Infatti, tali limitazioni dovrebbero valere, dato un certo problema, **per ogni algoritmo risolutivo che utilizzi una predefinita quantità di risorsa** e caratterizzare tutti questi algoritmi presenta una difficoltà enorme rispetto alla Teoria della Calcolabilità, dove invece si deve 'semplicemente' discriminare tra l'uso di un tempo finito o infinito.

2.3 La Macchina di Turing e le Classi di complessità:

La Macchina di Turing è un modello di calcolo capace di riconoscere parole su un alfabeto Σ. Più precisamente, un **alfabeto** Σ è un insieme di simboli e una **stringa** o **parola** su Σ è una sequenza finita di simboli giustapposti dell'alfabeto Σ. Se Σ è un alfabeto, Σ^* denota

la sua **chiusura**, ossia l'insieme di tutte le stringhe finite di simboli di Σ (inclusa la stringa vuota), mentre Σ^n denota l'insieme di tutte le stringhe di lunghezza n di simboli di Σ.

Definizione di Linguaggio:

Un Linguaggio (formale) L è un insieme di stringhe di lunghezza finita costruite sopra un alfabeto finito, cioè un sottoinsieme di Σ^*, $L \subseteq \Sigma^*$. Un linguaggio può essere definito tramite un modello formale che può generare o riconoscere tutte e sole le stringhe appartenenti al linguaggio stesso: a seconda del tipo di approccio, un modello si può definire in una grammatica formale (approccio generativo), o in un automa (approccio riconoscitivo).

Per poter risolvere un problema con una Macchina di Turing (MdT) è quindi necessario codificare le istanze del problema stesso con parole di Σ^*: un dato problema è allora estensivamente individuato dal linguaggio $L \subseteq \Sigma^*$ formato dalle parole che codificano istanze che verificano la questione posta dal problema.

Una **Macchina di Turing M deterministica** ad un nastro sull'alfabeto Σ è descritta da:

$$M = \langle S, A, \delta \rangle$$

dove:

(a) S è un insieme finito che identifica gli stati di controllo della macchina. S contiene almeno tre stati distinti s_0, q_{SI}, q_{NO}: s_0 è lo stato iniziale, q_{SI} e q_{NO} sono gli stati di arresto, rispettivamente 'accettante' e 'non accettante';

(b) A è insieme finito contenente Σ ed uno speciale simbolo β, indicatore di 'casella vuota';

(c) δ è detta 'funzione di transizione' della macchina.

$$\delta : A \times S \rightarrow A \times S \times \{1, 0, -1\}$$

La macchina M lavora su una memoria a nastro potenzialmente infinita, formata da celle numerate a partire da 1. Essa ha la sua unità di controllo ed una testina che collega l'unità di controllo al nastro. Inizialmente, la testina è posizionata sulla cella 1 e l'unità di controllo si trova nello stato s_0 e, se la parola di ingresso è $(x_1, x_2, ..., x_n)$, la cella i-esima contiene x_i, se $i \leq n$, altrimenti contiene β. La computazione procede per passi discreti. Supponiamo che, al passo K, lo stato sia S e la testina sia posizionata sulla cella p contenente a. Se $\delta(a, S) = (b, t, m)$, allora al passo $K+1$, lo stato di controllo è t, la cella p contiene b e la testina è posizionata in $p+m$. La computazione termina quando lo stato di controllo è q_{SI} o q_{NO}. Nel primo caso la parola $w = (x_1, x_2, ..., x_n)$ viene accettata, nel secondo caso essa viene respinta.

Il comportamento della MdT M è dato dal linguaggio riconosciuto da essa, L_M, che è l'insieme di tutte le stringhe sulle quali la macchina termina nello stato di accettazione q_{SI}.

Due risorse vengono usate come misura di complessità computazionale della MdT M e sono:

(a) Il tempo di calcolo $T_M(w)$;

(b) Lo spazio di memoria utilizzato $S_M(w)$.

Se l'ingresso è costituito dalla parola w, con $T_M(w)$ si denota il numero di passi di calcolo al termine della computazione, mentre con $S_M(w)$ si denota il massimo numero di celle usate durante la computazione.

Siamo ora in grado di denotare, fissato un limite di risorsa, i problemi risolubili entro tale limite di risorsa: le **Classi di complessità**.
Data una funzione $f : N \to N$ non decrescente, denotiamo:
$\mathbf{TIME}(f(n)) = \{L|\exists M(L = L_M, T_M(w) \leq f(|w|))\}$
$\mathbf{SPACE}(f(n)) = \{L|\exists M(L = L_M, S_M(w) \leq f(|w|))\}$
Classi di grande interesse sono la classe **P** dei problemi risolubili in tempo polinomiale da MdT deterministiche e la classe **PSPACE**, la classe dei problemi risolubili in spazio polinomiale da MdT deterministiche.
Formalmente:

$$\mathbf{P} = \bigcup_{k=0}^{\infty} TIME(n^k)$$
$$\mathbf{PSPACE} = \bigcup_{k=0}^{\infty} SPACE(n^k)$$

Vi è un sostanziale accordo nel considerare 'risolubili in modo efficiente' i problemi nella classe **P** (cfr. [Garey Johnson 1979]). Purtroppo, esiste una vasta classe di problemi di interesse pratico per cui non sono noti algoritmi risolutivi esatti che lavorino in tempo polinomiale. Questi problemi possono essere caratterizzati come i problemi risolubili in tempo polinomiale su modelli di calcolo non deterministico.
Un classico modello di calcolo non deterministico è dato dalla **Macchina di Turing non deterministica** (NMdT) (per dettagli si veda, ad esempio, [Garey Johnson 1979]). Essa può essere descritta da:

$$M = \langle S, A, \delta_1, \delta_2 \rangle$$

dove S ed A sono definiti come la macchina deterministica e, per $i \in 1, 2$:

$$\delta_i : A \times S \to A \times S \times \{1, 0, -1\}$$

Nel modello deterministico, ogni ingresso $w \in \Sigma^*$ induce una computazione, cioè una sequenza finita di configurazioni. Ogni configurazione è individuata dal contenuto del nastro, stato di controllo e posizione della testina.

Nel modello non deterministico, se al passo K la testina si trova in una configurazione C, al passo $K+1$ potrà trovarsi in una configurazione C' se si sceglie di eseguire δ_1, o in una configurazione C'' se si sceglie di eseguire δ_2. Data la parola di ingresso w, si portanno pertanto generare più computazioni: l'input è accettato se almeno una di esse termina nello lo stato di accettazione q_{SI}.

Data una NMdT M, il suo comportamento è il linguaggio L_M formato dalle parole accettate da M.

Se la parola w è accettata, con $T_M(w)$ si denota la lunghezza della più corta computazione accettante e con $S_M(w)$ il minimo numero di celle usato in una computazione accettante.

Data una funzione $f : N \to N$ non decrescente, si possono introdurre le seguenti Classi di complessità:

Classe NTIME($f(n)$) $= \{L | \exists NMdTM(L = L_M, w \in L_M, T_M(w) \leq f(|w|))\}$

Classe NSPACE($f(n)$) $= \{L | \exists NMdTM(L = L_M, w \in L_M, S_M(w) \leq f(|w|))\}$

Una classe che contiene molti problemi di grande interesse (dalla Logica alla Ricerca Operativa) è la classe **NP** dei problemi risolubili da una NMdT in tempo polinomiale.

Formalmente:

$$\mathbf{NP} = \bigcup_{k=0}^{\infty} NTIME(n^k)$$

Analogamente, porremo:

$$\mathbf{NPSPACE} = \bigcup_{k=0}^{\infty} NSPACE(n^k)$$

Quindi, **NPSPACE** è la classe dei problemi risolubili in spazio polinomiale dal modello non deterministico.
In una macchina non deterministica, per ogni configurazione C sono possibili due configurazioni prossime C' e C''. Se realizziamo una computazione assegnando come configurazione prossima C' con probabilità $p = \frac{1}{2}$ e C'' con probabilità $p = \frac{1}{2}$, otteniamo il modello di calcolo noto come **Macchina di Turing probabilistica** (MdTP).
In sostanza, una **Macchina di Turing probabilistica** (MdTP) è una Macchina di Turing non deterministica M con le seguenti caratteristiche:

(a) Ogni cammino non deterministico ha la stessa lunghezza e viene scelto con la stessa probabilità. La computazione è allora vista come un esperimento casuale, nel quale ogni uscita di un nodo decisionale (branch) ha uguale probabilità di essere scelta;

(b) Senza perdita di generalità, si suppone che tutti i nodi decisionali abbiano due uscite;

(c) Data una computazione realizzata da M su un ingresso x, poniamo $M(x) = 1$ se tale computazione termina in q_{SI}, $M(x) = 0$ se termina in q_{NO}.

Dalle definizioni poste, si vede che ogni computazione probabilistica di lunghezza t ha probabilità di essere scelta pari a 2^{-t}: quindi, il valore $M(x)$ di una computazione di M sull'input x viene ad essere una variabile casuale e pertanto si può valutare la seguente probabilità:

$$Prob\{M(x) = \beta\}$$

dove $\beta \in \{1, 0\}$.

Una classe di complessità di interesse per questa Tesi è la classe dei problemi risolubili in tempo polinomiale da una MdT probabilistica, senza limite di errore. Più precisamente, la classe **PP** [Gill 1977] è la classe dei linguaggi $L \subseteq \Sigma^*$ per cui esiste una macchina di Turing probabilistica M, con un limite polinomiale al numero di passi, tale che $\forall x \in \Sigma^*$,

$$Prob\{M(x) = X_L(x)\} > \frac{1}{2}$$

dove X_L è la funzione caratteristica del linguaggio L, cioè:

$$X_L(x) = 1 \text{ se } x \in L$$
$$X_L(x) = 0 \text{ se } x \notin L$$

Un ultimo modello di interesse, per la nostra trattazione, è quello di Macchina di Turing con Oracolo.

Una **Macchina di Turing con Oracolo** (OMdT) è una Macchina di Turing dotata dei seguenti elementi addizionali:

(a) Un linguaggio $A \subseteq \Sigma^*$, detto 'Insieme Oracolo';

(b) Un nastro aggiuntivo speciale, detto 'Nastro di Oracolo';

(c) Tre stati speciali: uno stato di domanda q_d e due stati di risposta q_S e q_N.

Il funzionamento di una OMdT è analogo a quello di una MdT tradizionale, con la differenza che quando la macchina è nella stato di domanda q_d viene letta la stringa y sul Nastro di Oracolo e lo stato successivo sarà:

$$q_S \text{ se } y \in A$$
$$q_N \text{ se } y \notin A$$

In entrambi i casi, la stringa y sul nastro di oracolo viene cancellata e la computazione procede dopo aver utilizzato l'informazione ottenuta dall'Oracolo (informazione ottenuta in maniera algoritmicamente gratuita e temporalmente istantanea). Data una classe di linguaggi C, con PP^C si intende la classe dei problemi decidibili da una MdT probabilistica che lavori in tempo polinomiale, disponendo di un Oracolo in C. Naturalmente, nulla vieta di 'gerarchizzare' i problemi in relazione alla possibilità di consultare, durante la computazione, un '**Oracolo**' e i problemi risolubili in tale modello saranno ovviamente strettamente dipendenti dal tipo di oracolo utilizzato: tale struttura può essere ricorsivamente utilizzata fino a creare una gerarchia di problemi relativizzati, usando ogni livello della gerarchia come oracolo per definire il livello successivo. In tal modo, si può definire la classe **CH** (Counting Hierarchy), introdotta nel 1986 da Wagner [Wagner 1986]:

$$C_0 = PP;$$
$$C_{k+1} = PP^{C_k}$$

La classe **CH** è allora definita da:

$$\mathbf{CH} = \cup_k^\infty \, C_k$$

2.4 Relazioni tra Classi di complessità e Problemi completi

Nel precedente paragrafo abbiamo introdotto alcune Classi di complessità:

(a) Le classi dei problemi risolubili in tempo polinomiale sul modello di Macchina di Turing deterministico **P**, non deterministico **NP** e probabilistico **PP**

(b) Le classi dei problemi risolubili in spazio polinomiale sul modello di Macchina di Turing deterministico **PSPACE** e non deterministico **NPSPACE**

(c) La classe **CH** derivata dalla 'Counting Hierarchy' (cfr. [Wagner 1986]).

Richiamiamo ora alcune relazioni note tra tali classi. Ovviamente (cfr. Papadimitriou [Papadimitriou 1993]) vale che $P \subseteq NP \subseteq PP \subseteq CH$ e $PSPACE \subseteq NPSPACE$; poichè un tempo polinomiale implica uno spazio polinomiale, sarà anche $P \subseteq PSPACE$.
Risultati meno immediati sono invece i seguenti, dovuti rispettivamente a Savitch [Savitch 1970] e a Wagner [Wagner 1986]:
Teorema di Savitch: NPSPACE = PSPACE.
Teorema di Wagner: CH $\subseteq$ PSPACE.

Otteniamo quindi le seguenti relazioni di inclusione tra le classi di nostro interesse:

$$P \subseteq NP \subseteq PP \subseteq CH \subseteq PSPACE = NPSPACE$$

Se queste inclusioni siano proprie o meno, è una problematica tutt'ora aperta: in particolare, decidere se $\mathbf{P} \neq \mathbf{NP}$ è considerato il più importante problema dell'Informatica Teorica ed uno tra i più importanti problemi matematici aperti. I tentativi di risolvere tale questione hanno tuttavia portato alla definizione di concetti importanti, quali quello di riducibilità polinomiale e di NP-completezza.

Definizione di Riducibilità Polinomiale: Dati due problemi $L_1, L_2 \subseteq \Sigma^*$, diremo che L_1 è polinomialmente riducibile (many-one) al problema L_2 (e scriveremo $L_1 < L_2$), se esiste una funzione $f : \Sigma^* \to \Sigma^*$ calcolabile in tempo polinomiale, tale che $x \in L_1 \Leftrightarrow$ f(x) $\in L_2$:

$$L_1 < L_2 \Leftrightarrow \exists\, f(f : \Sigma^* \to \Sigma^*, \text{ f calcolabile polinomialmente}, x \in L_1 \Leftrightarrow \text{f(x)} \in L_2)$$

È facile provare il seguente:

Teorema: se $L_1 < L_2$ e $L_2 \in P$, allora $L_1 \in P$.

Il significato intuitivo di tale risultato è che, se $L_1 < L_2$, allora L_2 è computazionalmente 'difficile' quanto L_1. Data una classe di complessità C, chiameremo 'Completi' i problemi 'più difficili' della classe (se esistenti). In particolare, per la classe **NP**:

Definizione di NP-completezza: Il problema L^* è **NP**-completo se:

(a) $L^* \in NP$;

(b) $\forall L \in NP \Rightarrow L < L^*$.

A partire dal noto risultato di Cook [Cook 1971], che prova l'esistenza di un problema **NP**-completo, è stata trovata una grande varietà di problemi **NP**-completi che sorgono naturalmente in vari settori dell'Informatica, della Ricerca Operativa e della Matematica Discreta. Tali

problemi sono i 'più difficili' della classe **NP**, dato che un qualuque altro problema **NP** si riduce polinomialmente ad uno di loro e quindi a tutti, dato che essi sono computazionalmente equivalenti fra loro: ciò implica, tra l'altro, che un eventuale algoritmo risolutivo polinomiale per uno solo di essi implicherebbe l'esistenza di un analogo algoritmo per tutti gli altri. Proprio l'assenza di una tale procedura, nonostante tutti gli sforzi compiuti, ha portato alla congettura che i problemi **NP**-completi non siano risolubili in tempo polinomiale e quindi non siano contenuti nella classe **P** (anche se finora nessuno ha dimostrato un tale risultato): vedi Garey-Johnson [Garey Johnson 1979].

2.5 Straight-line Program e Compressione aritmetica

In molte applicazioni è conveniente, ove possibile, non lavorare direttamente sui dati a disposizione ma su di una loro descrizione 'compressa'. 'Comprimere' qui significa dare una succinta descrizione dei dati in esame, descrizione che può essere poi ricostruita tramite un opportuno algoritmo di decompressione: come esempi si possono qui citare gli algoritmi di compressione di Huffman [Huffman 1952] o Lempel-Ziv [Lempel Ziv 1977], oltre alle ben note trasformazioni analogico-digitali e alle tecniche di campionamento e ricostruzione tramite filtraggio. La stessa Analisi delle Componenti Principali (PCA) [Pearson 1901, Jolliffe 2002], uno degli algoritmi di apprendimento non supervisonato più comuni, può essere visto come una tecnica di compressione lineare ottimale.

Forse la più nota interpretazione di compressione è la misura di complessità di Kolmogorov [Kolmogorov 1968]. In prima approssimazione, dato un linguaggio di programmazione, la complessità di Kolmogorov di una parola $x \in \{0,1\}^*$ è la lunghezza del più corto programma che, su un input ϵ, stampa x. Purtroppo, tale misura non è una funzione computabile; per le applicazioni, va in qualche modo 'semplificata' richiedendo, ad esempio, che il programma abbia solo istruzioni di assegnamento. Tale impostazione porta proprio a definire gli **Straight-line Programs** (SLP) e, in quest'ottica, comprimere significa dare un breve SLP che calcola il dato in esame. In questa Tesi daremo solo un accenno di tale tecnica, peraltro molto flessibile, rimandando alla letteratura per le definizioni formali [Claude 2008, Gasieniec 2003].
Benchè si possa definire un SLP su di una qualunque Algebra, per le nostre finalità ci limiteremo a lavorare sull'anello degli interi $\langle \mathbf{Z}, +, \cdot, -, 0, 1\rangle$ e sul campo dei razionali $\langle \mathbf{Q}, +, \cdot, -, /, 0, 1\rangle$:

Definizione: Uno **Straight-line Program** (SLP) Φ sull'anello $\langle \mathbf{Z}, +, \cdot, -, 0, 1\rangle$ con variabili $X_1, X_2, ..., X_n$ è una sequenza di n istruzioni, la cui la i^a istruzione è della forma:
$X_i = 0/1/X_j + X_s/X_j - X_s/X_j \cdot X_s$, con $0 \leq j, s < i$.
Dato un SLP Φ, ogni variabile X_k 'vale' un intero eval(X_k) definito induttivamente:

(a) Se $X_k = 0$, allora eval(X_k) = 0

(b) Se $X_k = 1$, allora eval(X_k) = 1

(c) Se $X_k = X_j + X_s$, allora eval(X_k) = eval(X_j) + eval(X_s)

(d) Se $X_k = X_j - X_s$, allora eval(X_k) = eval(X_j) - eval(X_s)

(e) Se $X_k = X_j \cdot X_s$, allora $\text{eval}(X_k) = \text{eval}(X_j) \cdot \text{eval}(X_s)$

Il valore denotato da Φ è l'intero $\text{eval}(\Phi)$, dove:

$$\text{eval}(\Phi) = \text{eval}(X_n)$$

In maniera perfettamente analoga, si definisce uno **Straight-line Program** (SLP) Φ sul campo dei razionali $\langle \mathbf{Q}, +, \cdot, -, /, 0, 1 \rangle$.
Al fine di chiarire meglio le definizioni introdotte e le potenzialità dell'uso degli SLP, consideriamo il seguente SLP Φ_n sull'anello $\langle \mathbf{Z}, +, \cdot, -, 0, 1 \rangle$:

$$X_0 = 1;\ X_1 = X_0 + X_0;\ X_2 = X_1 \cdot X_1;\ ....;\ X_n = X_{n-1} \cdot X_{n-1}.$$

È immediato verificare che tale SLP Φ_n calcola il numero $2^{2^{n-1}}$, che è rappresentabile in notazione binaria tramite $2^{n-1} + 1$ bit: pertanto, Φ_n costituisce una descrizione compressa (precisamente lineare in n) di un ente di grandezza esponenziale nella usuale rappresentazione binaria.
Un problema che si pone naturalmente, avendo uno SLP Φ, è capire se il numero da esso rappresentato sia non negativo. Formalmente:
Problema: PosSLP
Istanza: uno SLP Φ su $\langle \mathbf{Z}, +, \cdot, -, 0, 1 \rangle$
Questione: è l'intero $\text{eval}(\Phi) > 0$?

La difficoltà è legata al fatto che, come visto nell'esempio precedentemente illustrato, con un programma di dimensioni n si possono denotare interi che, rappresentati in binario, richiedono un numero esponenziale di bit (fino a 2^n). In tal senso, è fondamentale il risultato dovuto ad Allender e al. [Allender 2009]:
Teorema (Allender e al.): PosSLP $\in CH$.

Un altro problema che è stato provato essere in **CH** (Allender e al.) è il seguente:

Problema: SQRT-Sum

Istanza: numeri interi $0 < x_1 \leq x_2 \leq .... \leq x_n$, un intero H

Questione: è $\sum_i^n \sqrt{x_i} > H$?

La caratterizzazione della complessità di questo problema fu posta come problema aperto da [Garey Graham Johnson 1976], in relazione allo studio del Problema del Commesso Viaggiatore Euclideo. Nonostante l'apparente semplicità del problema, dopo molti anni di ricerche il miglior risultato lo pone in **CH** [ABKM 2006].

2.6 Complessità computazionale del Problema della Localizzazione del p-Centroide $(p - LC)$

Abbiamo visto che, dati numeri razionali $x_1, x_2, ..., x_n$, è univocamente definibile il loro centroide C_p rispetto ad una norma $||.||_p$, con $p \geq 1$ (cfr. **Appendice A**). In questo paragrafo studiamo la difficoltà computazionale della Localizzazione di C_p: dato un numero razionale h, è $C_p < h$?

Il problema può essere presentato formalmente come segue:

Problema: Localizzazione del p-Centroide (p-LC):

Istanza: numeri razionali $0 < x_1 \leq x_2 \leq .... \leq x_n$ ed un numero

razionale h

Questione: stabilire se $C_p < h$ dove:

$$C_p = arg\min_x \sum |x_i - x|^p$$

Come vedremo tra breve, il metodo di Compressione Aritmetica fornito dagli Straight-line Program appena introdotto ci consentirà di capire a quale livello delle gerarchia di complessità computazionale appartenga il Problema della Localizzazione del p-Centroide (p-LC): in particolare, proveremo che $p - LC \in CH$. Tale risultato non sembra facilmente migliorabile perchè ad esso, per $p = \frac{3}{2}$, si riduce il problema SQRT-Sum.

La tecnica che seguiremo per dimostrare che il Problema della Localizzazione del p-Centroide $\in CH$ sarà la seguente: per p generico, ridurremo il problema p-LC a PosSLP. Dato che già sappiamo essere PosSLP $\in CH$ [Allender 2009], avremo la limitazione inferiore di complessità cercata.

A questo scopo, introduciamo un risultato di gap (dovuto a Canny) [Canny 1988] che, in maniera del tutto generale, si applica alle soluzioni di un sistema di equazioni algebriche.

Teorema dello Scarto di Canny:

Siano $x_1, x_2, ..., x_N$ numeri che verificano un sistema algebrico di N equazioni in N incognite, avente un numero finito di soluzioni, con grado totale massimo d, con coefficienti interi relativi minori o uguali a M in valore assoluto. Allora, $\forall\ i = 1, ..., N$ si ha:

$$x_i = 0 \text{ oppure } |x_i| > \epsilon, \text{ dove } \epsilon = (3Md)^{-Nd^N}.$$

Questo teorema è molto utile in quanto fornisce un metodo per provare che un numero è zero: si calcola un intervallo che lo contiene sicura-

mente, di ampiezza minore di ϵ. Se l'intervallo non contiene lo zero, il numero è chiaramente diverso da zero e il suo segno è noto; altrimenti, se l'intervallo contiene lo zero ed ha ampiezza minore di ϵ, allora il numero in esame può essere soltanto uguale a zero.
Tornando al problema della localizzazione del p-Centroide, esso richiede di decidere, dati numeri razionali $x_1, x_2, ..., x_n; h$ se:

$$C_p = arg\min_x \sum |x_i - x|^p < h$$

Sappiamo già (cfr. **Appendici A** e **B**), che per $p > 1$ la funzione

$$y(x) = \sum_i^n |x_i - x|^p$$

è convessa e che quindi esiste un unico minimo C_p: pertanto, il problema è ben posto. Per $p = 1$ non vale in generale l'unicità, ma in questo caso possiamo porre $C_p = Mediana(x_1, x_2, ..., x_n)$. Senza perdere di generalità, possiamo supporre $x_1, x_2, ..., x_n, h$ numeri interi. Infatti, siano:

$$x_1 = \frac{a_1}{b_1}, x_2 = \frac{a_2}{b_2}, ..., x_n = \frac{a_n}{b_n}, h = \frac{a_{n+1}}{b_{n+1}}$$
$$\text{con } D = mcm\ (b_1, b_2, ..., b_n, b_{n+1})$$

Si verifica facilmente che la soluzione del problema su istanza $(x_1, x_2, ..., x_n, h)$ è la stessa sull'istanza di interi $(D \cdot x_1, D \cdot x_2, ..., D \cdot x_n, D \cdot h)$ poichè:

$$log_2\ D \leq \sum_1^{n+1} log_2\ (b_i + 1)$$

e quindi la dimensione (i. e. il numero di bit) dell'istanza $(D \cdot x_1, D \cdot x_2, ..., D \cdot x_n, D \cdot h)$ è polinomialmente legata alla dimensione dell'istanza $(x_1, x_2, ..., x_n, h)$.

Il principale risultato di questa sezione è il seguente:

Teorema 2.1: Per ogni $p > 1$, il Problema della Localizzazione del p-Centroide (p-LC) $\in CH$; se inoltre p è intero, allora $p - LC \in P$.

Dimostrazione: Per questioni di ordine e chiarezza, dividiamo la dimostrazione in due casi distinti:

(a) per p intero, il problema $\in P$;

(b) per p razionale, con una dimostrazione basata su una riduzione a PosSLP (ottenuta tramite il Teorema dello Scarto di Canny), vedremo che il problema $\in CH$ (Counting Herarchy).

2.6.1 Caso $p > 0$ intero: $p - LC \in P$.

Il caso p = 1 è semplice in quanto C_p viene ad essere la Mediana dei numeri $(x_1, x_2, ..., x_n)$: quindi, il problema è ridotto a controllare che

$$Med(x_1, x_2, ..., x_n) < h$$

Quindi, per risolvere il problema, è sufficiente il calcolo della mediana più 1 solo confronto; il calcolo della mediana di n numeri richiede $O(n)$ confronti (vedi ad es. [Vijay 2002]) e quindi $p - LC \in P$.
Se $p > 1$, si verifica, dalla convessità del funzionale $\sum |..|^p$, che:

$$C_p < h \Leftrightarrow \sum_{x_i<h}(h - x_i)^{p-1} - \sum_{x_i>h}(x_i - h)^{p-1} > 0$$

Il problema $p - LC$ può allora essere risolto controllando che:

$$\sum_{x_i<h}(h - x_i)^{p-1} - \sum_{x_i>h}(x_i - h)^{p-1} > 0$$

Tale verifica richiede 1 solo confronto, $2 \cdot n$ somme e n potenze (p-1)esime, ognuna delle quali richiede $O(log_2\, p)$ prodotti. Anche in questo caso, quindi, risulta che $p - LC \in P$. $\square$

2.6.2 Caso $p > 1$ razionale non intero: $p-LC \in CH$.

Come visto precedentemente, il problema si riduce a verificare che:

$$\sum_{x_i<h}(h - x_i)^{p-1} - \sum_{x_i>h}(x_i - h)^{p-1} > 0$$

Purtroppo, in questo caso i numeri $|x_i - h|^{p-1}$ non sono più interi, ma algebrici. Prima di proseguire nella dimostrazione del Teorema in esame, diamo il seguente **Lemma**, che ci dà un limite inferiore della complessità computazionale del problema che stiamo analizzando:

Lemma: 2.1 Il problema SQRT-Sum è polinomialmente riducibile a $\frac{3}{2}$-LC.

Dimostrazione: Sia data un'istanza $(A_1, A_2, ..., A_m; B_1, B_2, ..., B_s)$ del problema SQRT-Sum. Il problema richiede di determinare se:

$$\sqrt{A_1} + ... + \sqrt{A_m} > \sqrt{B_1} + ... + \sqrt{B_s}$$

Ora, all'istanza $(A_1, A_2, ..., A_m; B_1, B_2, ..., B_s)$ associamo l'istanza $(x_1, x_2, ..., x_m, x_{m+1}, ..., x_{m+s}; h)$ di $\frac{3}{2}$-LC, dove:

$$h = MaxA_j$$
$$x_i = h - A_i \text{ per } i \leq m$$
$$x_{m+j} = h + B_j \text{ per } 1 \leq j \leq s$$

La riduzione è chiaramente calcolabile in tempo polinomiale ed inoltre $(x_1, x_2, ..., x_{m+s}; h)$ verifica $\frac{3}{2}$-LC se e solo se $\sqrt{A_1} + ... + \sqrt{A_m} > \sqrt{B_1} + ... + \sqrt{B_s}$. □

Questo Lemma prova che SQRT-Sum è essenzialmente più 'facile' di $\frac{3}{2}$-LC. Trovare un risultato che localizzi $\frac{3}{2}$-LC in sottoclassi di **CH** (tipo **NP** o **P**), localizzerebbe automaticamente anche SQRT-Sum in tali classi, ottenendo un drastico miglioramento dopo svariati anni di ricerche.
Mostriamo ora, con una tecnica vicina a quella descritta in [ABKM 2006] che, per ogni p razionale, p-LC $\in CH$. L'idea è di osservare che il numero α:

$$\alpha = \sum_{x_i<h}(h - x_i)^{p-1} - \sum_{x_i>h}(x_i - h)^{p-1}$$

è un numero algebrico e quindi per esso vale il Teorema dello Scarto di Canny: se $p - 1 = \frac{a}{b}$, con a e b primi tra loro, il numero algebrico α è ottenibile come una combinazione lineare di $\sqrt[b]{M_k}$, com M_k interi: per determinare il segno di α basterà quindi approssimare numericamente, in maniera opportuna, i numeri $\sqrt[b]{M_k}$.
Pertanto, la questione se $C_p < h$ è ridotta a controllare che:

$$C_p < h \Leftrightarrow \sum_{x_i<h}(h - x_i)^{p-1} - \sum_{x_i>h}(x_i - h)^{p-1} > 0$$

cioè a controllare il segno di α:

$$\alpha = \sum_{x_i<h}(h - x_i)^{p-1} - \sum_{x_i>h}(x_i - h)^{p-1} = \sum_{x_i<h}\sqrt[b]{M_i} - \sum_{x_j>h}\sqrt[b]{M_j}$$

dove:

$$M_i = |h - x_i|^a$$

Ora, il numero α verifica il seguente sistema algebrico:

$$\begin{cases} \alpha = \sum_{x_i<h} Z_i - \sum_{x_i>h} Z_i \\ \mathrm{Z}_i^b = M_i \ (1 \leq i \leq n) \end{cases}$$

Sia $M = Max\ M_i$: per il Teorema dello Scarto di Canny, per determinare il segno di α basta approssimare α con un errore ϵ pari al più a $(3Mb)^{-nb^n}$. È quindi sufficiente approssimare Z_i con un errore δ uguale al più a $\frac{1}{n} \cdot (3Mb)^{-nb^n}$, $\forall i\ 1 \leq i \leq n$.
A tal fine, possiamo considerare il seguente SLP su campo Q dei razionali $\langle \mathbf{Q}, +, \cdot, -, / \rangle$, che fornisce una coincisa rappresentazione di $\sqrt[b]{M_i}$ utilizzando il Metodo di Newton (cfr. **Appendice C**):

$$\begin{gathered} X_0 = \lceil \sqrt[b]{M_i} \rceil; \\ X_1 = \frac{X_0 \cdot (b-1)}{b} + \frac{M_i}{(b \cdot X_0^{b-1})}; \\ ....; \\ X_k = \frac{X_{k-1} \cdot (b-1)}{b} + \frac{M_i}{(b \cdot X_{k-1}^{b-1})}. \end{gathered}$$

Poichè il Metodo di Newton comporta una convergenza quadratica (cfr. **Appendice C**), risulta:

$$|X_k - \sqrt[b]{M_i}| \leq 2^{-2^{O(k)}}.$$

Scegliendo k tale che $2^{-2^{O(k)}} = \delta$, noi possiamo approssimare $\sqrt[b]{M_i}$ con la precisione richiesta tramite un SLP di dimensione $O(n + log(M))$.
Pertanto, possiamo decidere se:

$$\sum_{i=1}^{k} (h - x_i)^{p-1} - \sum_{j=k+1}^{N} (x_j - h)^{p-1} > 0$$

cioè, equivalentemente:

$$C_p < h$$

controllando se la somma algebrica (con segni noti) di un numero polinomiale di numeri razionali M_i calcolabili come $M_i = eval(\Phi_i)$, dove Φ_i sono SLP (di dimensione polinomiale) su $\langle \mathbf{Q}, +, \cdot, -, / \rangle$, rappresenti un numero maggiore di $0 \in \mathbf{Q}$. Questo problema è riducibile a PosSLP, provando che ad ogni SLP Φ su $\langle \mathbf{Q}, +, \cdot, -, / \rangle$ possono essere associati (in tempo polinomiale) due SLP Π e Δ su $\langle \mathbf{Z}, +, \cdot, - \rangle$ tali che:

$$eval_{\mathbf{Q}}(\Phi) = \frac{eval_{\mathbf{Z}}(\Pi)}{eval_{\mathbf{Z}}(\Delta)}$$

Supponiamo infatti che Φ contenga M variabili $X_1, X_2, ..., X_M$: è possibile associare a Φ, in tempo polinomiale, un SLP Δ su $\boldsymbol{Z}$ $\langle \mathbf{Z}, +, \cdot, - \rangle$ con $2 \cdot N$ variabili $U_1, V_1, U_2, V_2, ..., U_N, V_N$ tali che:

$$eval(\Phi) = \frac{eval(U_N)}{eval(V_N)}$$

Infatti, per ogni variabile Z_k di Φ si possono associare le due variabili U_k, V_k di Δ tale che:

$$eval(Z_k) = \frac{eval(U_k)}{eval(V_k)}$$

Ad esempio, se $Z_k = Z_s + Z_l$, basta porre in Δ le due istruzioni:

$$U_k = U_s \cdot V_l + U_l \cdot V_s$$
$$V_k = U_s \cdot U_l$$

così che si abbia:

$$\frac{U_k}{V_k} = \frac{U_s}{V_s} + \frac{U_l}{V_l}$$

Analogamente, se $Z_k = Z_s \cdot Z_l$, basta porre:

$$U_k = U_s \cdot U_l$$
$$V_k = V_s \cdot V_l$$

mentre se $Z_k = \frac{Z_s}{Z_l}$:

$$U_k = U_s \cdot V_l$$
$$V_k = V_s \cdot U_l$$

Aggiungendo a Δ l'istruzione finale $Y = U_M \cdot V_M$, otteniamo così il programma Π su $\langle \mathbf{Z}, +, \cdot, - \rangle$. Si osserva:

$$eval(\Phi) = \frac{eval(U_N)}{eval(V_N)}$$
$$eval(\Pi) = eval(U_N) \cdot eval(V_N)$$

Ne segue che:

$$eval(\Phi) > 0 \text{ sse } eval(\Pi) > 0$$

Questo riduce, in tempo polinomiale, il problema p-LC a PosSLP: poichè già sappiamo essere PosSLP $\in CH$ [Allender 2009], si ha allora che anche p-LC $\in CH$. $\square$

Capitolo 3

Clustering vincolato in R^1: proprietà delle soluzioni ottime

3.1 Introduzione

Con 'Clustering vincolato' intendiamo il problema di Clustering in cui siano posti vincoli numerici sulle cardinalità delle classi nelle partizioni che rappresentano le soluzioni ammissibili del problema. La definizione formale del problema di Clustering vincolato verrà introdotta nel prossimo paragrafo e il resto del capitolo sarà dedicato allo studio delle proprietà delle soluzioni ottime del problema di Clustering vincolato, quando gli elementi da clusterizzare siano numeri razionali (Clustering vincolato in R^1).

Una nota proprietà delle soluzioni ottime del Clustering senza vincoli in R^1 è la cosiddetta **'String Property'**. Informalmente, una partizione

è contigua se le sue classi sono formate da numeri consecutivi: la 'String Property' asserisce che le soluzioni ottime di un problema di Clustering sono contigue. La 'String Property' fu dimostrata per la prima volta da Edwards e Cavalli-Sforza (1965) [Edwards e Cavalli-Sforza 1965] per il clustering in R^1 con norma L_2, ed è stata estesa da Novick (2009) [Novick 2009] al clustering in R^1 per ogni norma L_p. Questa proprietà delle soluzioni ottime riduce considerevolmente lo spazio di ricerca (con evidenti vantaggi computazionali), permettendo in alcuni casi di ottenere algoritmi notevolmente efficienti.

Il nostro contributo in questo capitolo sarà di mostrare che questa proprietà continua a valere anche per il Clustering vincolato, presentandola prima nel caso del Bi-clustering ed estendendola poi al caso generale. Questo risultato è stato fondamentale per lo sviluppo degli algoritmi particolarmente efficienti che saranno presentati nel **Capitolo 4**.

3.2 Clustering vincolato: definizioni preliminari

In generale, il problema di Clustering può essere formulato come segue. Si consideri uno spazio R^N dotato di metrica $||.||_p$, con $p \geq 1$. Dati n elementi $x_1, x_2, ..., x_n \in R^N$ e un intero $m > 1$, si vuole determinare una partizione $(A_1, A_2, ..., A_m)$ di $\{1, 2, ..., n\}$ che minimizzi

$$\sum_{i=1}^{m} \sum_{j \in A_i} ||x_j - C_{A_i}||_p^p$$

dove C_{A_i} è il p-centroide di A_i.

Si osservi che in questo caso le soluzioni ammissibili sono partizioni $(A_1, A_2, ..., A_m)$ di $\{1, 2, ..., n\}$, con l'unico vincolo di essere formate da m classi. Nel problema di Clustering Vincolato, il cui studio è oggetto di questa Tesi, si pongono dei vincoli ulteriori sulla cardinalità delle classi $A_1, A_2, ..., A_m$. Il problema può essere così formulato, dopo aver fissato una norma $||.||_p$ in R^N, con $p \geq 1$:

Problema: Clustering Vincolato (p-CV):
Istanza: n elementi $x_1, x_2, ..., x_n \in R^N$, un intero $m > 1$, un vettore $(k_1, k_2, ..., k_m)$ di m interi positivi (cardinalità dei cluster)
Risposta: una partizione $(A_1, A_2, ..., A_m)$ di $\{1, 2, ..., n\}$, con $|A_1| = k_1$, $|A_2| = k_2$, ..., $|A_m| = k_m$, che minimizza

$$\sum_{i=1}^{m} \sum_{j \in A_i} ||x_j - C_{A_i}||_p^p$$

dove C_{A_i} è il p-centroide di A_i.

Chiameremo infine Bi-clustering (Bi-clustering Vincolato) il problema riferito a istanze in cui $m = 2$; questo caso è particolarmente interessante, perchè una sua soluzione costituisce la base degli algoritmi gerarchici divisivi [Edwards e Cavalli-Sforza 1965, Scott e Symons 1971, Fisher 1958, Grower 1967].
Benchè si sia formulato il problema di Clustering vincolato per vettori di R^N, gran parte dei risultati presentati in questa Tesi si riferiscono al caso $N = 1$: gli elementi da clusterizzare sono in questo caso numeri razionali $x_1 \leq x_2 \leq .... \leq x_n$. L'interesse per lo studio di questo caso particolare è motivato da varie ragioni:

(a) Ai fini della comprensione della struttura dei problemi, è illuminante vedere cosa accade in spazi di bassa dimensione, soprattutto nel tentativo di generalizzare proprietà e strutture algoritmicamente utili;

(b) Il caso del Clustering in R^1 presenta delle caratteristiche peculiari (String property), caratteristiche che lo rendono di per se stesso interessante;

(c) In generale, anche il semplice problema del Bi-clustering per vettori arbitrari è NP-difficile [Drineas 2004, Aloise 2009]; il progetto di algoritmi esatti efficienti sarà possibile quindi solo per piccoli valori di N, ed in particolare per $N = 1$;

(d) Alcune applicazioni recenti, provenienti dal settore Bioinformatico, risultano modellizzabili in modo naturale come problemi di Clustering vincolato in R^1.

Preliminarmente al progetto di algoritmi risolutivi, è opportuno studiare alcune caratteristiche presenti nelle soluzioni ammissibili ottime. Un'importante caratteristica delle soluzioni ottime in R^1 è la cosiddetta **String property** [Vinod 1969].
Dati i numeri razionali $x_1 \leq x_2 \leq .... \leq x_n$, una partizione $(B_1, B_2, ..., B_m)$ di $\{1, 2, ..., n\}$ è detta **contigua** se per ogni s $(1 \leq s \leq m)$ e per ogni $i, j \in B_s$, vale che

$$\text{se } x_i < x_k < x_j, \text{ allora } x_k \in B_s.$$

La proprietà che ogni soluzione ottima del problema di Clustering sia contigua è nota in letteratura come **'String Property'**. Essa è stata dimostrata per la prima volta in [Edwards e Cavalli-Sforza 1965] nel

caso di norma L_2, ed è stata estesa in [Novick 2009] per tutte le norme L_p, con $p \geq 1$. Il principale risultato che otterremo in questo capitolo sarà l'estensione della 'String Property' anche al caso di Clustering vincolato. Nel prossimo paragrafo proveremo la 'String Property' nel caso del Bi-clustering vincolato: tale proprietà sarà poi successivamente dimostrata anche per il caso generale.

3.3 Bi-clustering vincolato in R^1 e 'String property'

Il problema di Bi-clustering vincolato è il problema di Clustering vincolato quando le soluzioni ammissibili sono partizioni composte da due sole classi.
In questo paragrafo studieremo alcune caratteristiche delle soluzioni ottime del Problema del Bi-clustering vincolato, mostrando che, per ogni norma L_p, con $p \geq 1$, almeno una soluzione ottima al problema verifica la 'String Property'. Nonostante la sua apparente semplicità, il caso $m = 2$ merita uno studio approfondito, sia a causa del suo interesse preliminare allo studio del caso più generale (qualche dimostrazione si farà per induzione ed è necessario che la proprietà sia esplicitamente vera nel caso base), sia perchè esso costituisce la base degli algoritmi gerarchici divisivi (cfr. [Edwards e Cavalli-Sforza 1965, Scott e Symons 1971, Fisher 1958, Grower 1967]).

Il problema di Bi-clustering vincolato, fissata un norma L_p, con $p \geq 1$, può essere descritto dal seguente problema di ottimizzazione:

Problema: Bi-Clustering Vincolato (p-2-CV)

Istanza: n vettori $x_1, x_2, ..., x_n \in R^N$; un intero $k < n$

Soluzioni ammissibili: Bipartizioni (A, B) di $\{1, 2, ..., n\}$, con $|A| = k$

Funzione obiettivo: $W(A, B) = \sum_{i \in A} ||x_i - C_A||_p^p + \sum_{j \in B} ||x_j - C_B||_p^p$, dove dove C_A e C_B sono i p-centroidi di A e B.

Tipo: Minimo.

Ogni soluzione ammissibile è quindi una bipartizione (A, B), con $|A| = k$ e $|B| = n - k$. Data un'istanza di (p-2-CV), indichiamo con (A^*, B^*) una soluzione ottima, tale cioè che:

$$\langle A^*, B^* \rangle = arg \min_{|A|=k} W(A, B)$$

Anche se il nostro interesse è legato al Bi-clustering vincolato in R^1, premettiamo un **Lemma** che stabilisce un'interessante proprietà per soluzioni ottime al Bi-clustering nel caso generale in cui $x_1, x_2, ..., x_n$ siano vettori di R^N.

Lemma 3.1: Dati $x_1, x_2, ..., x_n \in R^N$, si ponga, per ogni bipartizione (A, B) di $\{1, 2, ..., n\}$, $W(A, B) = \sum_{i \in A} ||x_i - C_A||_p^p + \sum_{j \in B} ||x_j - C_B||_p^p$. Sia (A^*, B^*) una partizione che minimizza $W(A, B)$ tra le partizioni in cui $|A| = k$, per un fissato $k < n$. Allora, per ogni $i \in A^*$ e $j \in B^*$:

$$||x_i - C_{A^*}||_p^p + ||x_j - C_{B^*}||_p^p \leq ||x_i - C_{B^*}||_p^p + ||x_j - C_{A^*}||_p^p$$

Dimostrazione: supponiamo per assurdo che esistano $i \in A^*$ e $j \in B^*$, tali che:

$$||x_i - C_{A^*}||_p^p + ||x_j - C_{B^*}||_p^p > ||x_i - C_{B^*}||_p^p + ||x_j - C_{A^*}||_p^p$$

Risulta allora:

$W(A^*, B^*) = \sum_{h\in A^*} ||x_h - C_{A^*}||_p^p + \sum_{h\in B^*} ||x_h - C_{B^*}||_p^p$
$= \sum_{h\in A^*-\{i\}} ||...||_p^p + \sum_{h\in B^*-\{j\}} ||...||_p^p + ||x_i - C_{A^*}||_p^p + ||x_j - C_{B^*}||_p^p$
$> \sum_{h\in A^*-\{i\}} ||...||_p^p + \sum_{h\in B^*-\{j\}} ||...||_p^p + ||x_i - C_{B^*}||_p^p + ||x_j - C_{A^*}||_p^p$
$= \sum_{h\in A^*-\{i\}\cup\{j\}} ||x_h - C_{A^*}||_p^p + \sum_{h\in B^*-\{j\}\cup\{i\}} ||x_h - C_{B^*}||_p^p$
$\geq \sum_{h\in A^*-\{i\}\cup\{j\}} ||x_h - C_{A^*-\{i\}\cup\{j\}}||_p^p + \sum_{h\in B^*-\{j\}\cup\{i\}} ||x_h - C_{B^*-\{j\}\cup\{i\}}||_p^p$
$= W(A^* - \{i\}\cup\{j\}, B^* - \{j\}\cup\{i\}).$

Questo è chiaramente assurdo perchè è $|A^* - \{i\}\cup\{j\}| = k$ e

$$W(A^* - \{i\}\cup\{j\}, B^* - \{j\}\cup\{i\}) < W(A^*, B^*)$$

mentre, per ipotesi, $\langle A^*, B^*\rangle$ è la soluzione ottima. □

Il penultimo passaggio è giustificato dalla definizione di p-Centroide (cfr. **Appendice B**), per cui risulta:

$\sum_{h\in A^*-\{i\}\cup\{j\}} ||x_h - C_{A^*}||_p^p \geq \sum_{h\in A^*-\{i\}\cup\{j\}} ||x_h - C_{A^*-\{i\}\cup\{j\}}||_p^p$

e

$\sum_{h\in B^*-\{j\}\cup\{i\}} ||x_h - C_{B^*}||_p^p \geq \sum_{h\in B^*-\{j\}\cup\{i\}} ||x_h - C_{B^*-\{j\}\cup\{i\}}||_p^p$

Una conseguenza del **Lemma 3.1** è il seguente:

Teorema 3.1: Nelle ipotesi di **Lemma 3.1**, l'insieme di vettori $\{x_i | i \in A^*\}$ è separato dall'insieme di vettori $\{x_j | j \in B^*\}$ da una superficie del tipo:

$$||x - \alpha||_p^p - ||x - \beta||_p^p = C$$

nel senso che, per $i \in A^*$ vale $||x_i - \alpha||_p^p - ||x_i - \beta||_p^p \leq C$, mentre per $j \in B^*$ vale $||x_j - \alpha||_p^p - ||x_j - \beta||_p^p \geq C$.

Dimostrazione: Da **Lemma 3.1** sappiamo che per ogni $i \in A^*$ e per ogni $j \in B^*$ vale:

$$||x_i - C_{A^*}||_p^p - ||x_i - C_{B^*}||_p^p \leq -||x_j - C_{B^*}||_p^p + ||x_j - C_{A^*}||_p^p$$

Quindi, posto:

$$C_1 = Max_{i \in A^*} \left\{ ||x_i - C_{A^*}||_p^p - ||x_i - C_{B^*}||_p^p \right\}$$
$$C_2 = Min_{j \in B^*} \left\{ -||x_j - C_{B^*}||_p^p + ||x_j - C_{A^*}||_p^p \right\}$$

Si ha:

$$C_1 \leq C_2$$
$$i \in A^* \Rightarrow \left\{ ||x_i - C_{A^*}||_p^p - ||x_i - C_{B^*}||_p^p \right\} \leq C_1$$
$$j \in B^* \Rightarrow C_2 \leq \left\{ -||x_j - C_{B^*}||_p^p + ||x_j - C_{A^*}||_p^p \right\}$$

Quindi, posto $\alpha = C_{A^*}$, $\beta = C_{B^*}$, $C = \frac{C_1 + C_2}{2}$, si ha la tesi. $\square$

Un'importante conseguenza del **Teorema 3.1** è il fatto che una soluzione ottima al Bi-clustering vincolato in R^1 verifica la 'String Property'. Vale in particolare:

Teorema 3.2: Fissato $p \geq 1$, siano assegnati n razionali $x_1 \leq x_2 \leq ... \leq x_n$ ed un intero $k < n$. Sia (A^*, B^*) una soluzione ottima al problema di Bi-clustering vincolato con norma L_p, fra le bipartizioni (A, B) con $|A| = k$. Allora (A^*, B^*) è una partizione contigua.

Dimostrazione: Distinguiamo il caso $p > 1$ dal caso $p = 1$.

Caso $p > 1$. Dal **Teorema 3.1** sappiamo che per una bipartizione ottima (A^*, B^*) esistono reali α, β, C tali che:

- $i \in A^* \Rightarrow |x_i - \alpha|^p - |x_i - \beta|^p \leq C$
- $j \in B^* \Rightarrow |x_j - \alpha|^p - |x_j - \beta|^p \geq C$

Sappiamo inoltre che $\alpha = C_{A^*}$, $\beta = C_{B^*}$, dove C_{A^*} e C_{B^*} sono i p-centroidi di A^* e B^*.

Consideriamo ora la funzione

$$f(x) = |x - \alpha|^p - |x - \beta|^p$$

Supponiamo che sia $\alpha > \beta$; in questo caso si ha che *f(x)* è monotona decrescente. Risulta infatti:

1. $f'(x) = p \cdot [(x - \alpha)^{p-1} - (x - \beta)^{p-1}]$ per $x > \alpha$
2. $f'(x) = -p \cdot [(\alpha - x)^{p-1} + (x - \beta)^{p-1}]$ per $\beta \leq x \leq \alpha$.
3. $f'(x) = p \cdot [-(\alpha - x)^{p-1} + (\beta - x)^{p-1}]$ per $x < \beta$

Si può quindi verificare, per ognuno dei tre intervalli, che $f'(x) < 0$ e quindi $f'(x) < 0$ per ogni valore di x. Vale inoltre:

- $\lim_{x \to +\infty} f(x) = -\infty$
- $\lim_{x \to -\infty} f(x) = +\infty$

Poichè $f(x)$ è continua, esiste un unico x^* per cui $f(x^*) = C$; inoltre:

- $i \in A^* \Rightarrow x_i \geq x^*$
- $j \in B^* \Rightarrow x_j \leq x^*$

Questo implica che la bipartizione (A^*, B^*) è contigua.
Supponiamo ora che sia $\alpha < \beta$. In base a un ragionamento analogo a quello precedentemente esposto, si prova che $f'(x) > 0$ per ogni x e quindi $f(x)$ è monotona crescente, ed inoltre si ha:

- $\lim_{x \to +\infty} f(x) = +\infty$
- $\lim_{x \to -\infty} f(x) = -\infty$

Poichè $f(x)$ è continua, esiste un unico x^* per cui $f(x^*) = C$. Quindi, anche in questo caso la bipartizione (A^*, B^*) risulta essere contigua.

Supponiamo infine che sia $\alpha = \beta$.

Se $x_1 = x_2 = ... = x_n$ la 'String Property' è banalmente verificata.

Ipotizziamo allora che sia $x_1 < x_n$ e consideriamo partizioni (A, B) rispettante i vincoli in cui, senza perdita di generalità, supponiamo che $1 \in A$.

Consideriamo, per ogni intero $s \geq 1$, il problema sui dati perturbati

$$x_1 - \tfrac{1}{s}, x_2, ..., x_n.$$

Data una partizione (A, B) con $1 \in A$, denotiamo con $W^{(s)}(A, B)$ il valore della funzione obiettivo su (A, B), con $C_A^{(s)}$ il p-centroide della classe A e con C_B il p-centroide della classe B, indipendente da s.

Se $t > s$ vale che $C_A^{(t)} > C_A^{(s)}$ (cfr. **Appendice A**), e quindi, per s sufficientemente grande (diciamo $s > s_0$) risulta che $C_A^{(s)} \neq C_B$ per ogni possibile partizione (A, B) rispettante i vincoli.

Per quanto detto nella parte iniziale della dimostrazione ($\alpha > \beta$ e $\alpha < \beta$), per $s > s_0$ esiste quindi una soluzione ottima contigua (A_s^*, B_s^*), tale che

$$W^{(s)}(A_s^*, B_s^*) \leq W^{(s)}(A, B)$$

per ogni (A, B) rispettante i vincoli.

Estraiamo una sottosequenza $S(j) \subseteq \mathbf{N}$ tale che $(A_{S(j)}^*, B_{S(j)}^*) = (A^*, B^*)$ per un'opportuna bipartizione contigua (A^*, B^*).

Poichè vale:

$$\lim_{s \to +\infty} W^s(A, B) = W(A, B)$$

si conclude che, per ogni bipartizione (A, B):

$$W(A, B) = \lim_{j \to +\infty} W^{S(j)}(A, B) \geq \lim_{j \to +\infty} W^{S(j)}(A^*, B^*) = W(A^*, B^*).$$

Abbiamo quindi identificato una bipartizione ottima contigua (A^*, B^*).

Il caso $p = 1$ può essere facilmente risolto per continuità su p. □

3.4 Clustering vincolato in R^1 e 'String property'

In questo paragrafo proveremo che una soluzione ottima al problema di Clustering vincolato in R^1, con partizioni formate da m classi ($m \geq 2$), verifica la **String Property**, estendendo alle soluzioni ottime del Clustering vincolato le proprietà osservato per il Bi-clustering. A tal fine, richiamiamo la definizione formale del problema di Clustering vincolato. Fissata una norma L_p con $p \geq 1$:

Problema: Clustering Vincolato (p-CV):

Istanza: n elementi $x_1, x_2, ..., x_n \in R^N$, un intero $m > 1$, un vettore $(k_1, k_2, ..., k_m)$ di m interi positivi (cardinalità dei cluster)

Risposta: Una partizione $(A_1, A_2, ..., A_m)$ di $\{1, 2, ..., n\}$, con $|A_1| = k_1$, $|A_2| = k_2$, ..., $|A_m| = k_m$, che minimizza

$$W(A_1, A_2, ..., A_m) = \sum_{i=1}^{m} \sum_{j \in A_i} ||x_j - C_{A_i}||_p^p$$

dove C_{A_i} è il p-centroide di A_i.

Supporremo, da ora in poi, senza perdita di generalità, che gli elementi $x_1, x_2, ..., x_n$ siano numeri razionali in R^1 ed inoltre siano $x_1 \leq x_2 \leq ... \leq x_n$.

Data la partizione $(A_1, A_2, ..., A_m)$ di $\{1, 2, ..., n\}$, diremo che la classe A_j è **consecutiva** se per ogni $i, k \in A_j$, se $x_i < x_s < x_j$, allora $s \in A_j$. Si osservi che una partizione $(A_1, A_2, ..., A_m)$ è **contigua** se e solo se tutte le sue classi sono consecutive.

Diremo che una partizione $(A_1, A_2, ..., A_m)$ è **estremale** se la classe contenente *1* è consecutiva. Infine, se la partizione $(A_1, A_2, ..., A_m)$ non è estremale ed A_j è la classe contenente *1*, definiamo

$$v(A_1, A_2, ..., A_m) = \min \{t | t \leq |A_j|, t \notin A_j\}$$

La definizione è ben posta poichè A_j non è consecutiva.

Il seguente **Lemma** implica che, per ogni partizione non estremale che verifica i vincoli, ne esiste una estremale verificante gli stessi vincoli.

Lemma 3.2: per ogni partizione $(A_1, A_2, ..., A_m)$ non estremale, esiste una partizione $(B_1, B_2, ..., B_m)$ con $|A_1| = |B_1|, |A_2| = |B_2|, ..., |A_m| = |B_m|$ e con $W(B_1, B_2, ..., B_m) \leq W(A_1, A_2, ..., A_m)$, tale che o $(B_1, B_2, ..., B_m)$ è estremale o $v(B_1, B_2, ..., B_m) > v(A_1, A_2, ..., A_m)$.

Dimostrazione: sia $t = v(A_1, A_2, ..., A_m)$ e $t \in A_s$. Per definizione, se A_j è la classe contenente *1*, vale che $j \neq s$. Consideriamo il problema di bipartizione vincolato su $\{x_i | i \in A_j \cup A_s\}$: per il **Teorema 3.2**, esiste una bipartizione ottima contigua (B_j, B_s) con $|A_j| = |B_j|$, $|A_s| = |B_s|$ e $W(B_j, B_s) \leq W(A_j, A_s)$.

Indichiamo con $(B_1, B_2, ..., B_m)$ la partizione ottenuta da $(A_1, A_2, ..., A_m)$ sostituendo A_j con B_j e A_s con B_s e lasciando immutate tutte le classi rimanenti. Ovviamente $|A_1| = |B_1|, |A_2| = |B_2|, ..., |A_m| = |B_m|$ e $W(B_1, B_2, ..., B_m) \leq W(A_1, A_2, ..., A_m)$.

Abbiamo due casi:

1. Caso: $1 \in B_j$. In questo caso, $t \in B_j$ per la contiguità della partizione (B_j, B_s), quindi o $(B_1, B_2, ..., B_m)$ è contigua o almeno $v(B_1, B_2, ..., B_m) > t = v(A_1, A_2, ..., A_m)$.

2. Caso: $1 \in B_s$. Abbiamo due possibilità:

   (a) Se $|B_s| = |A_s| = k_s \leq t$, allora $B_s = \{1, 2, ..., k_s\}$ e quindi $(B_1, B_2, ..., B_m)$ è una partizione estremale.

(b) Se $k_s > t$, allora $\{1, 2, ..., t\} \subseteq B_s$ e quindi o $(B_1, B_2, ..., B_m)$ è estremale o almeno $v(B_1, B_2, ..., B_m) > t = v(A_1, A_2, ..., A_m)$.

Osservando ora che $v(A_1, A_2, ..., A_m) \leq \text{Max } k_i$, applicando in modo iterativo il **Lemma 3.2** si ottengono i seguenti:

Corollario 3.1: Per ogni partizione $(A_1, A_2, ..., A_m)$ che verifica i vincoli $|A_1| = k_1$, $|A_2| = k_2$, ..., $|A_m| = k_m$, esiste una partizione estremale $(B_1, B_2, ..., B_m)$ che rispetta i vincoli (cioè $|B_1| = k_1$, $|B_2| = k_2$, ..., $|B_m| = k_m$) ed inoltre

$$W(B_1, B_2, ..., B_m) \leq W(A_1, A_2, ..., A_m).$$

Corollario 3.2: Fra le partizioni $(A_1, A_2, ..., A_m)$ ottime che verificano i vincoli $|A_1| = k_1$, $|A_2| = k_2$, ..., $|A_m| = k_m$ ne esiste una estremale.

Siamo ora in grado di ottenere il principale risultato di questa sezione, che estende la 'String Property' per alcune soluzioni ottime del problema di Clustering vincolato. Vale infatti:

Teorema 3.3: Dati n razionali $x_1 \leq x_2 \leq ... \leq x_n \in R^1$ ed un vettore $(k_1, k_2, ..., k_m)$ di m interi positivi, con $\sum_{i=1}^{m} k_i = n$, esiste una soluzione ottima $(A_1^*, A_2^*, ..., A_m^*)$ al problema di Clustering vincolato tale che la partizione ottima $(A_1^*, A_2^*, ..., A_m^*)$ è contigua.

Dimostrazione: Per induzione su m.

Per $m = 2$, la proprietà è vera per il **Teorema 3.2**.

Supponendo che la proprietà sia vera per *m-1*, dimostriamo che vale per m.

Sia $x_1, x_2, ..., x_n$ e $(k_1, k_2, ..., k_m)$ una istanza del problema di Clustering vincolato in R^1 e supponiamo che $(C_1, C_2, ..., C_m)$ sia una partizione ottima. Ciò

significa che, per ogni partizione ottima $(A_1, A_2, ..., A_m)$ che rispetta i vincoli $|A_1| = k_1$, $|A_2| = k_2$, ..., $|A_m| = k_m$, vale:

$$W(C_1, C_2, ..., C_m) \leq W(A_1, A_2, ..., A_m).$$

Ricordiamo qui che:

$$W(A_1, A_2, ..., A_m) = \sum_{k=1}^{m} W(A_k).$$

dove $W(A_k) = \sum_{i \in A_k} |x_i - C_{A_k}|^p$, essendo C_{A_k} il p-centroide della classe A_k.
Per il **Corollario 3.2** sappiamo che esiste una partizione estremale $(B_1, B_2, ..., B_m)$ che rispetta i vincoli (cioè $|B_1| = k_1$, $|B_2| = k_2$, ..., $|B_m| = k_m$), e per cui

$$W(B_1, B_2, ..., B_m) \leq W(C_1, C_2, ..., C_m).$$

Sia B_j la classe consecutiva cui appartiene *1.* Consideriamo il problema di Clustering vincolato in R^1 che ammette come istanza $\{x_i | i \leq \cup_{s \neq j} B_s\}$ e con *m-1* vincoli $(k_1, k_2, ..., k_{j-1}, k_{j+1}, ..., k_m)$.
Per ipotesi di induzione esiste una soluzione ottima e contigua $(A_1^*, ..., A_{j-1}^*, A_{j+1}^*, ..., A_m^*)$ per cui:

$$W(A_1^*, ..., A_{j-1}^*, A_{j+1}^*, ..., A_m^*) \leq W(B_1, ..., B_{j-1}, B_{j+1}, ..., B_m)$$

Vale allora:

$$\begin{aligned} & W(A_1^*, ..., A_{j-1}^*, B_j, A_{j+1}^*, ..., A_m^*) \\ = & W(A_1^*, ..., A_{j-1}^*, A_{j+1}^*, ..., A_m^*) + W(B_j) \\ \leq & W(B_1, ..., B_{j-1}, B_{j+1}, ..., B_m) + W(B_j) \\ = & W(B_1, ..., B_{j-1}, B_j, B_{j+1}, ..., B_m) \end{aligned}$$

Poichè per ipotesi $(B_1, B_2, ..., B_m)$ è una soluzione ottima, si conclude che $(A_1^*, ..., A_{j-1}^*, B_j, A_{j+1}^*, ..., A_m^*)$ è una partizione ottima e contigua per il problema in esame. □

Capitolo 4

Clustering vincolato in R^1: algoritmi e applicazioni

4.1 Introduzione

Nel precedente capitolo abbiamo osservato che il Clustering vincolato in R^1 ammette soluzioni ottime che risultano essere partizioni contigue; limitare lo spazio di ricerca a partizioni contigue introduce forti semplificazioni al problema, portando al progetto degli algoritmi che saranno illustrati in questo capitolo.

In primo luogo, viene affrontato il problema del Bi-clustering vincolato: esso consiste nel calcolo della funzione obiettivo su due sole partizioni, dando luogo ad algoritmi esatti (nel caso di $||.||_1$ e $||.||_2$) o approssimati (nel caso generale) particolarmente efficienti. Si ricorda che il problema del Bi-clustering in R^N è invece NP-difficile.

Per quanto riguarda il problema di Clustering vincolato in m classi, la validità della String Property riduce la ricerca della soluzione ottima al calcolo del valore della funzione obiettivo su $m!$ partizioni: l'estensione al caso generale

dell'algoritmo del Bi-clustering porta dunque ad algoritmi che lavorano, nel caso peggiore, in tempo esponenziale. Tale situazione non può essere migliorata: nel terzo paragrafo proveremo infatti che, per ogni $||.||_p$, il problema del clustering vincolato in R^1 è **NP**-difficile, e diventa **NP**-completo nel caso di $||.||_1$ o $||.||_2$.

Viene poi studiata una versione rilassata del problema del Clustering vincolato. Nel clustering in m classi, con vincoli di cardinalità $(k_1, k_2, ..., k_m)$, le soluzioni ammissibili sono partizioni $(A_1, A_2, ..., A_m)$, con $|A_1| = k_1$, $|A_2| = k_2$, ..., $|A_m| = k_m$; nella versione rilassata, le soluzioni ammissibili sono invece partizioni $(A_1, A_2, ..., A_m)$ tali che per ogni i, $|A_i| \in \{k_1, ..., k_s\}$: cioè i cluster possono avere cardinalità non rigorosamente prefissate, ma appartenenti ad un certo insieme. Per il problema del Clustering vincolato rilassato in R^1 si è progettato un efficiente algoritmo, basato su tecniche di programmazione dinamica.

Infine, viene presentata una potenziale applicazione dell'algoritmo realizzato (per il Clustering vincolato rilassato in R^1) ad uno specifico problema in ambito bioinformatico: in particolare si osserva che, a un certo livello di approssimazione, l'identificazione di regioni promotore in sequenze genomiche può essere modellata come un problema di Clustering con vincoli in R^1.

4.2 Algoritmi e complessità

4.2.1 Algoritmo per il Bi-Clustering vincolato

Fissata un norma L_p, con $p \geq 1$, il problema di Bi-clustering vincolato richiede, dati n numeri razionali $x_1, x_2, ..., x_n$ e un intero $k < n$, di determinare una bipartizione (A^*, B^*) di $\{1, 2, ..., n\}$, con $|A^*| = k$, che minimizzi

$$W(A, B) = \sum_{i \in A} |x_i - C_A|^p + \sum_{j \in B} |x_j - C_B|^p$$

Sappiamo dal **Teorema 3.2** che il Bi-clustering vincolato R^1 verifica la 'String Property, cioè che ammette una soluzione ottima (A^*, B^*) contigua. Fissata la cardinalità k di A^*, esistono solo 2 bipartizioni contigue:

$$(\{1, 2, ..., k\}, \{k+1, ..., n\}),$$
$$(\{n-k+1, ..., n\}, \{1, ..., n-k\}),$$

Per la 'String Property, basta scegliere quale di queste due partizioni è la migliore. Ciò da luogo al seguente:

ALGORITMO: Alg BCV

Input: (razionali $x_1 \leq x_2 \leq ... \leq x_n$; intero $k < n$)

$a = W(\{1, 2, ..., k\}, \{k+1, ..., n\})$;

$b = W(\{n-k+1, ..., n\}, \{1, ..., n-k\})$;

Return:

if $a < b$ **then** return $(\{1, 2, ..., k\}, \{k+1, ..., n\})$

else return $(\{n-k+1, ..., n\}, \{1, ..., n-k\})$

Alg BCV risolve correttamente il problema di Bi-clustering vincolato utilizzando un numero polinomiale di operazioni di somma e prodotto, più un confronto, tra numeri algebrici. Per una realizzazione corretta si pongono quindi problemi simili a quelli analizzati in **Capitolo 2**, in relazione al problema 'Localizzazione del p-Centroide': rimane quindi un problema aperto se sia possibile realizzare una implementazione di Alg BCV in tempo polinomiale.

È naturalmente possibile lavorare con 'approssimazioni razionali' dei numeri algebrici. In questo caso è possibile, data una bipartizione (A, B), calcolare in tempo polinomiale $O(m \cdot n)$ un'approssimazione $W^*(A, B)$ di $W(A, B)$ tale che:

$$|W^*(A,B) - W(A,B)| \leq 2^{-n}.$$

L'algoritmo che si ottiene non è generalmente corretto (il test $a < b$ può fornire una risposta errata), ma da un lato lavora in tempo polinomiale e dall'altro garantisce che il valore della funzione obiettivo nella risposta fornita differisca dall'ottimo di al più 2^{-n}.
Nei casi importanti di norme L_1 o L_2, Alg BCV risulta invece corretto lavorando in tempo polinomiale.

4.2.2 Esempi

In questi esempi, fissata la norma $||.||_p$ e data una sequenza $x_1 \leq x_2 \leq ... \leq x_n$ di interi, calcoliamo, per ogni k $(1 \leq k \leq \frac{n}{2})$, il valore della soluzione ottima $W_p(k)$ al Bi-clustering vincolato su istanza $x_1, x_2, ..., x_n$ e k che è il minimo tra i due valori:

$$W_p(\{1,2,...,k\},\{k+1,...,n\})$$
$$W_p(\{n-k+1,...,n\},\{1,...,n-k\})$$

1. Esempio: sequenza $x_i = i \cdot \lceil log(i) \rceil$ con $1 \leq i \leq 100$

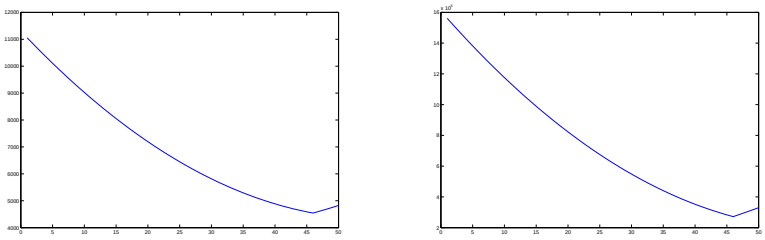

Figura 4.1: sequenza $x_i = i \cdot \lceil log(i) \rceil$, (s) norma L_1 (d) norma L_2

2. Esempio: sequenza $x_i = i^2$ con $1 \leq i \leq 100$

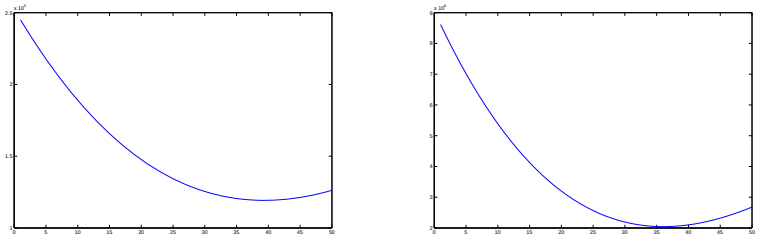

Figura 4.2: sequenza $x_i = i^2$, (s) norma L_1 (d) norma L_2

3. Esempio: sequenza $x_i = i^5$ con $1 \leq i \leq 100$

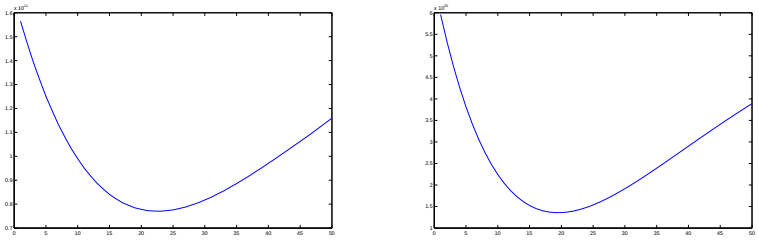

Figura 4.3: sequenza $x_i = i^5$, (s) norma L_1 (d) norma L_2

Considerazioni:

Si osservi che i grafici delle funzioni obiettivo (vedi da Figura 4.1 a Figura 4.3), per norme con $p \in [1, 2]$, risultano essere piuttosto regolari, dando luogo a curve unimodali. Un simile andamento può essere compreso ossevando la notevole regolarità delle successioni in istanza.

Le conclusioni che si possono empiricamente trarre, supportate anche dallo studio analitico del caso $p = 2$ (l'unico caso in cui si hanno formule esplicite per il calcolo del centroide e della funzione obiettivo, coincidenti rispettivamente con la media e la varianza moltiplicata per la cardinalità del cluster), sono che per generatori con sequenze regolari di tipo n^l, la funzione $W_p(k)$ è unimodale, presentando un unico minimo locale. Si può inoltre osservare una tendenza alla regolarizzazione delle funzioni obiettivo all'aumentare di p.

Si pone a questo punto la questione di costruire la sequenza x_i (non decrescente) per cui la curva $W_p(k)$ presenti più minimi. L'idea è considerare due sequenze crescenti $f(j)$ e $g(j)$; dato n, fissare l'istanza:

$$(f(1), ..., f(1), f(2), ..., f(2), f(3), ...f(3), ..., f(n), ..., f(n))$$
$$\text{con } f(1) \text{ ripetuto } g(n) \text{ volte;}$$
$$\text{con } f(2) \text{ ripetuto } g(n-1) \text{ volte;}$$
$$....$$
$$\text{con } f(n) \text{ ripetuto } g(1) \text{ volte;}$$

Nel seguente esempio, poniamo $f(j) = g(j) = 2^j$ ed $n = 100$ (vedi Figura 4.4). Per tale istanza calcoliamo $W_1(k)$, $W_2(k)$ e $W_4(k)$; come si può osservare in Figure 4.4 e 4.5, le curve $W_1(k)$ e $W_2(k)$ presentano numerosi minimi locali; la curva $W_4(k)$ (vedi Figura 4.5) ritorna ad essere unimodale.

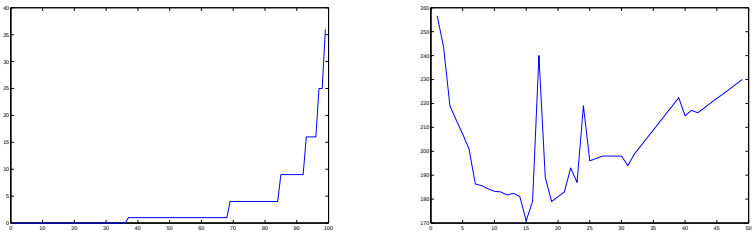

Figura 4.4: (s) sequenza x_i = crescita esponenziale a tratti,(d) norma L_1

4.2.3 Clustering vincolato è NP-difficile

Fissata una norma L_p, con $p \geq 1$, il Problema del Clustering vincolato può essere formalmente descritto dal seguente problema di ottimizzazione:

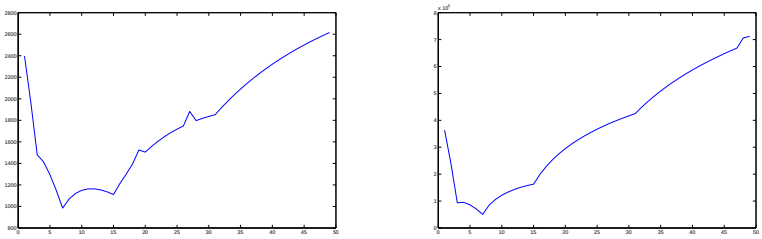

Figura 4.5: (s) norma L_2,(d) norma L_4

Problema: Problema del p-Clustering vincolato (p-CV)
Istanza: numeri razionali $x_1 \leq x_2 \leq ... \leq x_n$; un $(k_1, k_2, ..., k_m)$ vettore di m interi positivi (cardinalità dei cluster), con $\sum_{i=1}^{m} k_i = n$
Soluzioni ammissibili: partizioni $(A_1, A_2, ..., A_m)$ di $\{1, 2, ..., n\}$ con:

$$|A_1| = k_1,\ |A_2| = k_2,\ ...,\ |A_m| = k_m$$

Funzione obiettivo: $W(A_1, A_2, ..., A_m) = \sum_{i=1}^{m} \sum_{j \in A_i} ||x_j - C_{A_i}||_p^p$ dove C_{A_i} è il p-centroide di A_i.
Tipo: Minimo.

Questo problema è la naturale estensione del Bi-Clustering vincolato, quando si prendono in considerazione partizioni formate da un generico numero m di classi. In base ai risultati di **Capitolo 3**, possiamo effettuare la ricerca della soluzione ottima fra le partizioni $(A_1, A_2, ..., A_m)$ che sono contigue e verificano

$$|A_1| = k_1,\ |A_2| = k_2,\ ...,\ |A_m| = k_m.$$

Purtroppo, il numero di tali soluzioni è $m!$ e quindi, per valori anche non troppo elevati di m, l'algoritmo che estende quello presentato nel precedente paragrafo risulta inefficiente, lavorando in tempo esponenziale.

In questo paragrafo proviamo che la versione di decisione del problema p-CV è **NP**-difficile: questo esclude (per $\mathbf{P} \neq \mathbf{NP}$) la possibilità di risolvere p-CV con algoritmi polinomiali esatti, anche quando $p = 1$ o $p = 2$. A questo proposito, la versione di decisione del problema di ottimizzazione è la seguente:

Problema: Problema p-CV (versione di decisione)
Istanza: numeri razionali $x_1 \leq x_2 \leq ... \leq x_n$; vettore $(k_1, k_2, ..., k_m)$ di m interi positivi (cardinalità dei cluster), con $\sum_{i=1}^{m} k_i = n$; razionale r
Questione: esiste una partizione $(A_1, A_2, ..., A_m)$ di $\{1, 2, ..., n\}$ con:

$$|A_1| = k_1,\ |A_2| = k_2,\ ...,\ |A_m| = k_m$$

tale che:

$$W(A_1, A_2, ..., A_m) = \sum_{i=1}^{m} \sum_{j \in A_i} ||x_j - C_{A_i}||_p^p < r?$$

dove C_{A_i} è il p-centroide di A_i.

Per provare che p-CV è **NP**-difficile basta ridurre polinomialmente a p-CV il problema 3-Partition, che è noto essere **NP**-completo.
Il problema 3-Partition è formalmente definito come segue:
Problema: 3-Partition
Istanza: interi $p_1, p_2, ..., p_{3 \cdot m}$
Questione: esiste una partizione $(S_1, S_2, ..., S_m)$ di $\{1, 2, ..., 3 \cdot m\}$ tale che, posto $B = \frac{\sum_{k=1}^{3 \cdot m} S_k}{m}$, sia:

$$B = \sum_{i \in S_1} S_i = \sum_{i \in S_2} S_i = ... = \sum_{i \in S_m} S_i?$$

Nel 1975 questo problema fu provato essere **NP**-completo da Garey e Johnson [Garey Johnson 1975]; il problema 3-partition rimane **NP**-completo anche nel caso in cui i numeri costituenti l'istanza siano limitati da un polinomio

in m.

Proviamo ora che $3 - Partition < p - CV$, per ogni $p \geq 1$.

Teorema 4.1: $\forall p \geq 1$, 3-Partition è polinomialmente riducibile a p-CV.

Dimostrazione: All'istanza $p_1, p_2, ..., p_{3 \cdot m}$ di 3-partition si associ, in tempo polinomiale, l'istanza di p-CV data da:

- interi $\langle (1, 1, ..., 1, 2, 2, ..., 2, ..., m, m, ..., m) \rangle$ (tutti i numeri uguali sono ripetuti ciascuno B volte)
- interi positivi $p_1, p_2, ..., p_{3 \cdot m}$
- un razionale r_p, con $r_p < (\frac{1}{2})^p$

Supponiamo che esista una partizione $(S_1, S_2, ..., S_m)$ tale che:

$$B = \sum_{i \in S_1} S_i = \sum_{i \in S_2} S_i = ... = \sum_{i \in S_m} S_i$$

È allora possibile trovare una partizione $(A_1, A_2, ..., A_{3 \cdot m})$ di $\{1, 2, ..., B \cdot m\}$ tale che, per ogni k, esiste un $f_k \in \{1, 2, ..., m\}$ per cui $j \in A_k \Rightarrow x_j = f_k$. Gli elementi del cluster A_k sono indici che denotano lo stesso valore numerico f_k. Il centroide C_{A_k} è quindi f_k; pertanto la funzione obiettivo è tale che:

$$W(A_1, A_2, ..., A_{3 \cdot m}) = \sum \sum |f_k - f_k|^p = 0$$

Supponiamo ora che si riesca a trovare una partizione $(A_1, A_2, ..., A_{3 \cdot m})$ con $|A_i| = p_i$ e tale che

$$W(A_1, A_2, ..., A_{3 \cdot m}) < r_p$$

In questo caso, per ogni k, il cluster individuato da A_k è formato da un unico elemento. Se così non fosse, per almeno un valore di k il cluster individuato da A_k dovrebbe contenere due elementi interi α e β distinti. Sarebbe allora:

$W(A_1, A_2, ..., A_{3\cdot m}) = \sum\sum|...|^p \geq \sum_{j\in A_k}|x_j - C_{A_k}|^p \geq |\alpha - C_{A_k}|^p + |\beta - C_{A_k}|^p$

Poichè $|\alpha - \beta| \geq 1$, $\text{Max}\{|\alpha - C_{A_k}|, |\beta - C_{A_k}|\} \geq \frac{1}{2}$ e questo permette di concludere:

$$W(A_1, A_2, ..., A_{3\cdot m}) \geq (\tfrac{1}{2})^p \geq r_p$$

Tale risultato è chiaramente assurdo. Poichè i cluster individuati da A_k, per ogni k, sono formati da un unico elemento, questo comporta che l'istanza $p_1, p_2, ..., p_{3\cdot m}$ di 3-partition ammette soluzione. □

Osserviamo ora che, per norme L_1 e L_2, data una soluzione ammissibile $(A_1, A_2, ..., A_m)$ per una istanza del problema di Clustering vincolato e dato un intero λ, è facile verificare se $W(A_1, A_2, ..., A_m) < \lambda$: basta eseguire un numero polinomiale di operazioni di somma, prodotto o confronto di numeri razionali. Questa osservazione, unitamente a **Teorema 4.1**, porta al seguente:

Corollario 4.1: 1-CV e 2-CV sono problemi **NP**-completi.

4.2.4 Algoritmo per il Clustering vincolato rilassato

La NP-Completezza del problema p-CV (per esempio per $p = 1$ o $p = 2$) impedisce di trovare algoritmi efficienti esatti (a meno che $\mathbf{P} = \mathbf{NP}$). Introduciamo ora una versione rilassata del problema di Clustering vincolato che può essere risolta efficientemente con tecniche di programmazione dinamica. Fissata una norma L_p, con $p \geq 1$, abbiamo:

Problema: p-Clustering Vincolato Rilassato (p-CVR)
Istanza: numeri razionali $x_1 \leq x_2 \leq ... \leq x_n$; un insieme $\{k_1, ..., k_s\}$ di

s interi positivi (possibili cardinalità dei cluster); un intero m (numero dei cluster)

Soluzioni ammissibili: partizioni $(A_1, A_2, ..., A_m)$ di $\{1, 2, ..., n\}$ tali che per ogni i, $|A_i| \in \{k_1, ..., k_s\}$

Funzione obiettivo: $W(A_1, A_2, ..., A_m) = \sum_{i=1}^{m} \sum_{j \in A_i} |x_j - C_{A_i}|^p$ dove C_{A_i} è il p-centroide di A_i.

Tipo: Minimo.

Mentre il problema p-CV è **NP**-difficile (e 1-CV e 2-CV sono **NP**-completi), la sua versione rilassata p-CVR può essere risolta da un efficiente algoritmo dinamico che, nel caso di $p = 1$ o $p = 2$, lavora in tempo polinomiale.

A questo riguardo, consideriamo una istanza di p-CVR composta dai razionali $x_1 \leq x_2 \leq ... \leq x_n$, dagli interi $\{k_1, ..., k_s\}$ e dall'intero m. Denotiamo con D la matrice di m righe e n colonne, dove $D(k, j)$ è il valore ottimale della funzione obiettivo calcolata raggruppando in k cluster i primi j elementi dell'istanza, cioè è il peso della soluzione ottima al problema p-CVR su istanza formata dai razionali $x_1 \leq x_2 \leq ... \leq x_j$, dagli interi $\{k_1, ..., k_s\}$ e dall'intero k (numero di cluster). Se per tale sottoproblema non esiste alcuna soluzione ammissibile, si pone $D(k, j) = +\infty$.

Denotiamo infine con C la matrice di n righe e n colonne, tale che:

$$C(i, j) = \begin{cases} \sum_{k=i}^{j} |x_k - \mu|^p \text{ se } |j - i| \in \{k_1, ..., k_s\} \\ +\infty \text{ altrimenti.} \end{cases}$$

In altri termini, $C(i, j)$ è il valore della funzione obiettivo calcolata sul cluster formato dagli elementi dell'istanza nelle posizioni consecutive $\{i, i + 1, i + 2, ..., j\}$.

L'algoritmo è basato sull'osservazione che:

- $D(1, j) = C(1, j)$ per $1 \leq j \leq n$)

- $D(k,j) = Min_t \{D(k-1, j-k_t) + C(j-k_t, j)\}$ per $k > 1$

Nella formula precedente si suppongono verificati i controlli di validità sugli indici delle matrici. La matrice $D(m,n)$ costituisce l'output della procedura: il generico elemento $D(k,j)$ è il valore ottimale della funzione obiettivo calcolata raggruppando in k cluster i primi j elementi dell'istanza e quindi la procedura fornisce i valori ottimali di tutte le partizioni formate dai primi j elementi in k cluster (con $j \in [1,n]$ e $k \in [1,m] \bigcap \{k_1, ..., k_s\}$). Il listato in pseudocodice del programma che trova gli m cluster minimi usando la programmazione dinamica è il seguente:

procedura [D] = mclusterdinamico (v,m,$\{k_1, ..., k_s\}$**)**
v = vettore degli elementi da clusterizzare, ordinato in senso crescente;
m = numero di cluster;
$\{k_1, ..., k_s\}$ = insieme delle cardinalità ammissibili dei cluster;

```
  Inizializzazione della matrice C
  for i = 1 : n do
    for j = 1 : n do
      if (i > j) then
        C(i,j) ← +∞
      else
        if se indici i e j validi then
          C(i,j) ← valore funzione obiettivo calcolata sugli elementi {v(i), ..., v(j)}
        else
          C(i,j) ← +∞
        end if
      end if
    end for
  end for
```

Inizializzazione della matrice D delle funzioni obiettivo

for j = 1 : n **do**
  $D(1, j) \leftarrow C(1, j)$
end for
for i = 2 : m **do**
  for j = 1 : n **do**
    $D(i, j) \leftarrow +\infty$
  end for
Calcolo della matrice D delle funzioni obiettivo
  for i = 2 : m **do**
    for j = 1 : n **do**
      for t = 1 : n **do**
        if indice t valido **then**
          $cand \leftarrow D(i-1, j-t) + C(j-t+1, j)$
          if $cand < D(i, j)$ **then**
            $D(i, j) \leftarrow cand$
          end if
        end if
      end for
    end for
  end for
Fine procedura

La complessità di tale algoritmo è di $O(m{\cdot}n^3)$ operazioni di somma, prodotto e confronto di numeri algebrici: in aggiunta, per ogni elemento $C(i, j)$ della matrice C, vi è il calcolo (e la memorizzazione) del valore della funzione obiettivo valutata sugli elementi $\{v(i), ..., v(j)\}$. Negli importanti casi di norme L_1 e L_2, l'algoritmo lavora in tempo polinomiale.

4.2.5 Ottimizzazione dell'algoritmo nel caso di norma L_2

In questo paragrafo svilupperemo una versione ottimizzata dell'algoritmo precedente, nel caso particolare in cui si adoperi la norma L_2: in tale situazione, infatti, si hanno delle formule chiuse che ci consentono di calcolare esplicitamente, dato il cluster, sia il centroide (che in questo caso particolare coincide con la media dei valori costituenti il cluster) sia il valore della funzione obiettivo (coincidente con la varianza moltiplicata per la cardinalità del cluster). Queste formule ci consentono di evitare di calcolare preliminarmente l'intera matrice $C(i,j)$, il cui costo computazionale (nel caso della norma L_2) è dell'ordine di $O(n^3)$. Infatti, anzichè calcolare l'intera matrice $C(i,j)$, calcoliamo preliminarmente i due vettori Q e S di dimensioni n:

$$Q(i) = \sum_{j=1}^{i} x_j^2$$
$$S(i) = \sum_{j=1}^{i} x_j$$

Il tempo di calcolo per ottenere Q e S è $O(n)$, osservando che essi possono essere ottenuti in maniera incrementale. In formule:

$$Q(i) = Q(i-1) + x_i^2$$
$$S(i) = S(i-1) + x_i$$

È immediato verificare che:

$$C(i,j) = \sum_{k=i}^{j} |x_k - \mu|^2 = (Q(j) - Q(i-1)) - \frac{(S(j)-S(i-1))^2}{j-i+1}$$

Nell'algoritmo ottimizzato il calcolo di $C(i,j)$, dati i vettori Q e S, richiede quindi solo un numero costante di operazioni aritmetiche, e questo consente di evitare il calcolo della matrice $C(i,j)$ (di dimensioni $n \times n$) e la relativa occupazione di memoria, velocizzando notevolmente la procedura. Il listato in pseudocodice del programma ottimizzato è il seguente:

procedura [D] = mclusterdinamico2 (v,m,$\{k_1, ..., k_s\}$)

v = vettore degli elementi da clusterizzare, ordinato in senso crescente;

m = numero di cluster;

$\{k_1, ..., k_s\}$ = insieme delle cardinalità ammissibili dei cluster;

Inizializzazione dei vettori *Q* e *S*

$Q(1) \leftarrow x_1^2$

$S(1) \leftarrow x_1$

for i = 2 : m **do**

$Q(i) \leftarrow Q(i-1) + x_i^2$

$S(i) \leftarrow S(i-1) + x_i$

end for

Inizializzazione della matrice D delle funzioni obiettivo

for j = 1 : n **do**

$D(1, j) \leftarrow Q(j) - \frac{S(j)^2}{j}$

end for

for i = 2 : m **do**

for j = 1 : n **do**

$D(i, j) \leftarrow +\infty$

end for

Calcolo della matrice D delle funzioni obiettivo

for i = 2 : m **do**

for j = 1 : n **do**

for t = 1 : n **do**

if indice t valido **then**

$cand \leftarrow D(i-1, j-t) + (Q(j) - Q(j-t)) - \frac{(S(j)-S(j-t))^2}{t}$

if $cand < D(i, j)$ **then**

$D(i, j) \leftarrow cand$

end if

end if

end for

end for

end for

Fine procedura

Il numero di operazioni aritmetiche dell'algoritmo ottimizzato è $O(n^2 \cdot m)$. Per una stima empirica della complessità dell'algoritmo, utile per prevedere il tempo necessario per il calcolo dei cluster nelle applicazioni reali, si è ipotizzata una complessità del tipo:

$$T = \alpha \cdot n^{\beta} \cdot m^{\gamma}$$

Per le stime di α e β, si è fissato $m = 10$ e posto $\gamma \approx 1$. Si è applicato l'algoritmo **mclusterdinamico2** a vettori casuali, le cui componenti erano numeri generati uniformemente in $[0, 1]$, di dimensioni tra 1000 e 2000 elementi (con passo 100): ogni esperimento è stato ripetuto 30 volte. Detto $T(n)$ il tempo di esecuzione su istanza $v(n)$ di dimensione n, è stata calcolata la retta di regressione dei punti $(log(n), log(T))$. Il valore stimato di α e β sui 30 esperimenti (con un livello di confidenza del $\frac{999}{1000}$) è stato di:

$$\alpha = (3.2 \mp 0.2) \cdot 10^{-7}$$

$$\beta = (1.95 \mp 0.5)$$

Questo porta a considerare il tempo di esecuzione per istanze di dimensione n, per $m = 10$, pari a:

$$T = 3.2 \cdot 10^{-8} \cdot n^{1.95} \cdot m$$

Una analoga stima per il coefficiente γ ha portato a:

$$\gamma = (1.05 \mp 0.01)$$

Il tempo di esecuzione su dati di dimensione n, per cui sia fissato il numero m di cluster, può quindi essere approssimativamente stimato da:

$$T = 3.2 \cdot 10^{-8} \cdot n^{1.95} \cdot m^{1.05}$$

Questi tempi sono stati ottenuti tramite implementazione del programma in Matlab 7.10 (2010a) su modello di calcolo HP PAVILION ELITE HPE, con 16 Gb di Ram e singolo processore da 2,8 GHz.

4.3 Potenziali applicazioni in ambito Bioinformatico

In questo paragrafo si mostra come problemi di notevole interesse bioinformatico, quali quello dell'identificazione di regioni promotore in sequenze genomiche [Shmid 2007], possano essere ragionevolmente modellati come problemi di Clustering vincolato in R^1 e risolti tramite algoritmi di clustering rilassato.

4.3.1 Regolazione della trascrizione e promotori

In primo luogo, si introducono alcune nozioni di base relativamente al processo di trascrizione e alla funzione dei promotori. Negli organismi superiori tutta l'informazione necessaria alla sintesi dei componenti di base di cellule, organi ed apparati risiede nel nucleo cellulare, in una lunga molecola di DNA i cui tratti biologicamente e funzionalmente significativi sono detti **geni**.

Trascrizione e traduzione. L'informazione codificata nel DNA come sequenza di nucleotidi (Adenina, Guanina, Timina, Citosina e Uracile) viene utilizzata per sintetizzare le proteine tramite i processi di *trascrizione* e di *traduzione*. Molto sinteticamente, il processo di trascrizione consiste nella copiatura dei geni in corrispondenti sequenze di mRNA (RNA

messaggero), la cui funzione è di trasmettere l'informazione ai ribosomi, dove le triplette di DNA verranno tradotte (processo di traduzione) in sequenze di aminoacidi (proteine) con il tramite del tRNA (RNA di trasporto, il cui compito consiste proprio nel trasporto degli aminoacidi).

Promotori. L'informazione contenuta nei geni è "densa": essa viene letta da appositi enti cellulari (i ribosomi), seguendo le regole del codice genetico in modo da produrre la proteina (che costituisce il prodotto codificato dal gene). È stato quindi necessario, nel corso dell'evoluzione, trovare nel genoma uno spazio esterno al gene in cui "conservare" l'informazione necessaria ai processi di regolazione del gene (motivi regolatori): dato che l'informazione contenuta nel genoma è in forma lineare, e quindi essenzialmente unidimensionale (il DNA è una lunga sequenza di caratteri), la maggior parte dei geni è regolata da una sequenza di circa 1000 lettere (nucleotidi), ubicata immediatamente a monte del gene che regolano. Tale regione è detta **promotore prossimale**, o più semplicemente **promotore** e contiene densi cluster di motivi regolatori. L'identificazione della regione promotore permette di effettuare analisi in grado di fornire sia indicazioni sulla funzione del gene che informazioni sui meccanismi molecolari alla base della sua regolazione.

4.3.2 Identificazione automatica delle regioni promotore mediante tecniche di clustering monodimensionale

L'identificazione automatica dei promotori è un'attiva area di ricerca in Bioinformatica, la quale negli ultimi anni ha acquisito un'importanza sempre maggiore a causa della drastica riduzione dei tempi di sequenziamento del genoma (attualmente ridotto ad alcuni mesi). Come già detto, poichè la struttura informativa esistente a livello sub-cellulare è intrinsecamente lineare, ciò permette di affrontare il problema dell'iden-

tificazione dei promotori in maniera naturale mediante l'applicazione di tecniche di clustering monodimensionale. Infatti, i dati biomolecolari grezzi sono costituiti da una stringa composta da 4 tipi di caratteri (a, t, c e g) rappresentante la regione genomica di interesse G e da una collezione di stringhe T (anche esse composte da caratteri appartenenti al medesimo alfabeto della sequenza genomica) rappresentanti, ognuna, la sequenza di un trascritto.

Preprocessing dei dati ed applicazione di algoritmi di clustering monodimensionali vincolati

Il primo passo dell'analisi è costituito dall'applicazione di un algoritmo per l'allineamento delle sequenze dei trascritti alla regione genomica. Poichè i trascritti che si vogliono identificare (quelli originati dalla trascrizione dei geni presenti nella regione genomica di interesse) sono stati originati da un processo di trascrizione di regioni (geni) presenti nella stessa regione genomica, il problema può essere modellato, in maniera molto spontanea e naturale, come una ricerca di stringhe occorrenti all'interno di una stringa di lunghezza molto maggiore. Questo particolare problema rappresenta quindi un caso speciale del problema dell'allineamento di biosequenze, problema in cui si cercano copie esatte di una stringa all'interno di un'altra stringa. Dato che la sequenza genomica costituisce un sistema di coordinate lineari e discrete (definite dalla posizione dei caratteri che costituiscono la sequenza), l'identificazione del sottoinsieme T_m delle stringhe appartenenti a T e presenti in G e la definizione delle loro coordinate rispetto a G (effettuato mediante metodi di allineamento) produce un file strutturato come l'esempio che segue:

```
query:  AK312116.1
target:  human chromosome 21

00000001 ttgcgtcgtagtctcctgcagcgtctggggtttccgttgcagtcctcgga 00000050
```

```
>>>>>>>> |||||||||||||||||||||||||||||||||||||||||||||||||| >>>>>>>>
33032004 ttgcgtcgtagtctcctgcagcgtctggggtttccgttgcagtcctcgga 33032053

00000051 accaggacctcggcgtggcctagcgagttatggcgacgaaggccgtgtgc 00000100
>>>>>>>> |||||||||||||||||||||||||||||||||||||||||||||||||| >>>>>>>>
33032054 accaggacctcggcgtggcctagcgagttatggcgacgaaggccgtgtgc 33032103

00000101 gtgctgaagggcgacggcccagtgcagggcatcatcaatttcgagcagaa 00000150
>>>>>>>> |||||||||||||||||||||||||||||||||||||||||||||||||| >>>>>>>>
33032104 gtgctgaagggcgacggcccagtgcagggcatcatcaatttcgagcagaa 33032153

00000151 g 00000151
>>>>>>>> | >>>>>>>>
33032154 g 33032154
```

Esso indica che il trascritto AK312116.1 è stato mappato sulla sequenza del cromosoma umano 21 e che il primo carattere del trascritto (indice 00000001) mappa in corrispondenza del carattere 33032004 della sequenza del cromosoma. Dato che qui si è interessati alla ricerca dei promotori che sono definiti in termini relativi al primo carattere del trascritto (regione di circa 1000 nucleotidi immediatamente a monte del primo carattere del trascritto), il dato utile è costituito solo dalla coordinata genomica (coordinata rispetto a G) del primo carattere del trascritto (nella fattispecie 33032004).

Una volta mappati tutti i trascritti appartenenti a T rispetto al sistema di coordinate di G, si ottiene una collezione di valori corrispondenti alle coordinate genomiche degli allineamenti di ognuno dei trascritti [1]

```
84046690 84046736 84046905 84046906 84046908 84046915 84046926
84046926 84046927 84046928 84046928 84046928 84046928 84046929
84046936 84046936 84046936 84046937 84046939 84046940 84046949
```

[1] Più precisamente i valori corrispondono alle coordinate genomiche dell'allinemento del primo nucleotide di ognuno dei trascritti rispetto alla sequenza genomica di riferimento G.

```
84046959 84046963 84046971 84046973 84046973 84046973 84046973
84046973 84046973 84046973 84046973 84046974 84046976 84046976
...
```

Ovviamente, i valori possono essere presenti più volte, in quanto è possibile che, nella collezione T, siano presenti sia copie multiple del medesimo trascritto che trascritti differenti che condividono la posizione del primo nucleotide. A questo punto, l'insieme di questi valori costituisce i dati in ingresso per una ricerca automatizzata delle posizioni dei promotori: infatti, tali dati sono interpretabili come punti (posizioni degli allineamenti) lungo una retta (la sequenza genomica di riferimento). È ora possibile applicare un algoritmo di **clustering monodimensionale con vincoli** che permetta di identificare degli "hotspot" di inizio della trascrizione corrispondenti all'ultimo nucleotide delle regioni promotore. Ad esempio, un ipotetico task sperimentale per l'identificazione automatica dei promotori, basato sul metodo precedentemente illustrato, potrebbe essere il seguente:

```
DATI GREZZI:

- sequenza G rappresentante una regione genomica
- collezione T di stringhe  rappresentanti  tutti
  i trascritti presenti in cellula

PREPROCESSING:

- mapping  delle   coordinate   di  ogni  elemento
  Ti in  termini di  coordinate in G

INPUT:

- collezione delle coordinate  genomiche dei primi
  nucleotidi dei trascritti mappabili nella regio-
```

```
  ne genomica in esame
- numero dei promotori/cluster m che si prevede ot-
  tenere

VINCOLI:

- numero minimo di punti presenti in un cluster
- numero massimo di punti presenti in un cluster

OUTPUT:

- insieme di m clusters/promotori
```

La validazione dei risultati, effettuata tramite l'utilizzo di un browser genomico, conclude la procedura proposta.

4.4 Setup sperimentale

Gli esperimenti proposti si pongono come obiettivo quello di testare la capacità dell'algoritmo sviluppato nell'ambito di questa Tesi di dottorato di riconoscere i settori del genoma dove si localizzano i punti di inizio della trascrizione genica. Per ottenere questo risultato, si sono utilizzati dati pubblici [Kim 2005]. Un esperimento simile a quello proposto, ma basato su metodi computazionali diversi, è stato recentemente pubblicato in [Shmid 2007]. In questo esperimento, gli autori hanno esaminato una regione del cromosoma 12 umano ad alta densità genica, avente un'estensione di circa 250 kilobasi: gli autori hanno analizzato dati ottenuti mediante una tecnica nota come 'immunoprecipitazione cromatinica' accoppiata a rilevazione di dati di espressione via microarray *ChIP-on-chip*, in modo da ottenere un miglior filtraggio del rumore trascrizionale.

Schematicamente, la tecnologia *ChIP-on-chip* permette di rilevare una serie di picchi caratterizzati da volume, estensione e centro, correlati direttamente sia alla frequenza di trascrizione dei geni che alla posizione dei punti in termini di coordinate genomiche. In [Shmid 2007] gli autori propongono un metodo di clustering mono dimensionale, MADAP, basato su misture di gaussiane e algoritmi di Expectation-Maximization, in grado di riconoscere le posizioni del genoma caratterizzate da elevata attività trascrizionale ed indicativi della presenza di un gene che viene trascritto nelle condizioni sperimentali in cui vengono misurati i dati di espressione. I risultati presentati in [Shmid 2007] dimostrano, mediante proiezione delle posizioni dei cluster ottenuti sulle coordinate genomiche (e successiva visualizzazione di queste ultime nel browser genomico EnsEMBL (http://www.ensembl.org/), che MADAP è effettivamente in grado di riconoscere regioni genomiche contenenti l'inizio di geni trascrizionalmente attivi.
In questo paragrafo, ci si è proposti di valutare l'applicabilità dell'algoritmo di clustering monodimensionale sviluppato, effettuando un esperimento simile a quello presentato in [Shmid 2007] al fine di poter confrontare i relativi risultati ottenuti.

4.4.1 Dataset

I dati di espressione rilevati mediante *ChIP-on-chip* nella regione genomica analizzata in [Shmid 2007] (cromosoma 12, posizione 6767416-6990346), sono stati ottenuti dal sito web del metodo MADAP (www.isrec.isb-sib.ch/madap/). Si è scelto di utilizzare i dati forniti dal sito del progetto MADAP in quanto essi sono stati preprocessati e l'utilizzo diretto dei dati di input utilizzati in [Shmid 2007] assicura l'assenza di bias nel confronto dei risultati ottenuti.
Di seguito, si da un tipico esempio di dati *ChIP-on-chip* disponibili in formato General Feature Format (GFF):

```
0       ##gff-version   3
1       ##remapped and transformed ChIP-chip data (GEO: GSE2672)CHR12 pos 6767416-6990346(hg18)
chr12   GEO:GSE2672     TSS     6768452 6768502 2       +       .       probeID:GPL2077_139738
chr12   GEO:GSE2672     TSS     6768752 6768802 8       +       .       probeID:GPL2077_139741
chr12   GEO:GSE2672     TSS     6768952 6769002 7       +       .       probeID:GPL2077_139743
chr12   GEO:GSE2672     TSS     6769052 6769102 5       +       .       probeID:GPL2077_139744
chr12   GEO:GSE2672     TSS     6769152 6769202 1       +       .       probeID:GPL2077_139745
...
```

In esso è possibile notare i seguenti elementi: il cromosoma (chr12), il codice che permette di estrarre il dataset dalla banca dati NCBI GEO (GEO:GSE2672), il tipo di feature genomica (TSS) e le sue coordinate (start, end). Ad ogni rilevazione è associato un valore di intensità (2 nel caso del primo record, 8 per il secondo) che rappresenta l'intensità del fenomeno di trascrizione verificatosi alle coordinate indicate.

4.4.2 Pre-processing dei dati

In primo luogo, onde evitare la rilevazione di eventi di trascrizione poco informativi, come nel caso dell'ultimo record presente nell'esempio di file GFF appena presentato (rilevato con intensità 1), si è deciso di filtrare, secondo quanto riportato in [Shmid 2007], i dati in modo da eliminare tutte le rilevazioni aventi un'intensità (cioè un livello di espressione) minore di *10*. A questo punto, avendo tutte le trascrizioni con intensità pari o superiore a *10*, si è creata un'istanza che tenga conto dell'intensità trascrizionale rilevata, duplicando la coordinata genomica tante volte quanto era il suo livello di espressione. Ad esempio, supponendo di dover filtrare i seguenti dati imponendo una intensità minima di segnale pari a *3*:

```
...
chr12   GEO:GSE2672     TSS     6775572 6775622 1       +       .       probeID:GPL2077_139779
chr12   GEO:GSE2672     TSS     6778208 6778258 1       +       .       probeID:GPL2077_139790
chr12   GEO:GSE2672     TSS     6779692 6779742 3       +       .       probeID:GPL2077_139796
chr12   GEO:GSE2672     TSS     6787531 6787581 1       +       .       probeID:GPL2077_139803
chr12   GEO:GSE2672     TSS     6787927 6787977 1       +       .       probeID:GPL2077_139804
chr12   GEO:GSE2672     TSS     6788027 6788077 2       +       .       probeID:GPL2077_139805
chr12   GEO:GSE2672     TSS     6789614 6789664 1       +       .       probeID:GPL2077_139812
```

```
chr12   GEO:GSE2672     TSS     6792124 6792174 3       +       .       probeID:GPL2077_139821
...
```

Dopo il filtraggio rimangono ovviamente solo i record n *3* e n *8*:

```
...
chr12   GEO:GSE2672     TSS     6779692 6779742 3       +       .       probeID:GPL2077_139796
chr12   GEO:GSE2672     TSS     6792124 6792174 3       +       .       probeID:GPL2077_139821
...
```

Duplicando le coordinate delle regioni genomiche associate alla lettura del dato un numero di volte pari al valore di intensità rilevato (nella fattispecie *3*), si ha l'istanza:

```
6779692
6779692
6779692
6792124
6792124
6792124
```

4.4.3 Scelta dei parametri

Naturalmente, il metodo proposto è stato sviluppato in modo da favorirne la massima generalità di utilizzo, anche se è tuttavia necessario tener presente che, per la sua efficace applicazione in ambiti di ricerca specifici (come la Bioinformatica), si rende necessaria una conoscenza a priori sul problema che permetta la selezione ottimale dei parametri da utilizzare in fase di sperimentazione.

Numero massimo di cluster selezionati:

In [Shmid 2007] gli autori sono riusciti ad identificare, nel test con cui ci si è confrontati, *8* cluster corrispondenti ad altrettanti siti trascrizionalmente attivi. Tenuto conto dell'inerente difficoltà nell'identificazione di regioni trascrizionalmente attive e considerando l'elevata densità genica nella regione genomica in esame, si è deciso empiricamente di variare i parametri del metodo proposto in modo da ottenere soluzioni composte da un numero massimo di *20* cluster. Tale scelta non è arbitraria perchè

in primo luogo non è restrittiva, in quanto l'algoritmo sviluppato fornisce **tutte** le soluzioni ottime comprese tra *1* e *20* cluster, e poi è giustificata dall'interesse per l'eventuale localizzazione di possibili nuovi geni non ancora annotati all'epoca in cui furono pubblicati i risultati presentati in [Shmid 2007].

Cardinalità dei cluster:

Per poter effettuare un confronto significativo con i risultati di [Shmid 2007], la cardinalità minima dei cluster cercati è stata posta uguale a *10*.

Scelta della norma:

Nonostante il metodo proposto sia in grado di operare con valori arbitrari della norma, risulta estremamente complesso trovare una giustificazione strettamente biologica per la scelta di quest'ultima. In assenza di tali specifiche informazioni, si è scelto di utilizzare la norma L_2, comunemente utilizzata in molti esperimenti descritti in letteratura. Inoltre, ciò ha consentito l'utilizzo dell'algoritmo ottimizzato **mclusterdinamico2** che, come già illustrato in un precedente paragrafo di questo capitolo, presenta una particolare efficienza computazionale.

4.4.4 Valutazione delle performance

L'analisi dei risultati sperimentali è stata condotta esaminando la capacità del metodo proposto di identificare geni trascrizionalmente attivi nella regione genomica di interesse. Le coordinate delle regioni genomiche nelle quali è attesa la presenza di un gene trascrizionalmente attivo sono state identificate tramite una finestra (di raggio τ) centrata sulla coordinata genomica del primo elemento di ogni cluster: in formule [start point cluster $_i - \tau + 1$,start point cluster $_i + \tau$] in cui τ è stato settato arbitrariamente a *750*. Tale valore permette di ottenere regioni di

1500 nucleotidi di ampiezza, compatibili con la dimensione di *1000-2000* nucleotidi dei promotori genici.
Successivamente, le regioni genomiche identificate dai cluster sono state confrontate posizionalmente con le annotazioni dei geni contenuti nella regione genomica analizzata ottenute direttamente dal database MySQL su cui è basato il browser genomico UCSC genome browser (http://genome.ucsc.edu/). Ogni volta che l'inizio di un gene risulta compreso nelle regioni genomiche identificate dai cluster ottenuti, si è considerato positivo l'esito della predizione.
Per poter correttamente confrontare i risultati prodotti dal metodo sviluppato con quelli ottenuti da MADAP, si sono utilizzate le annotazioni geniche disponibili al momento della pubblicazione di [Shmid 2007]. È tuttavia opportuno tenere presente che l'annotazione dei geni è un processo continuo nel tempo ed è quindi corretto, in presenza di eventuali falsi positivi, effettuare un confronto delle regioni genomiche in esame anche con le annotazioni resesi disponibili dopo la pubblicazione del lavoro usato come termine di paragone.

4.5 Risultati

In questo paragrafo, dedicato alla presentazione dei risultati sperimentali, si descrive il comportamento del metodo per l'identificazione di siti genomici trascrizionalmente attivi, utilizzando dati *ChIP-on-chip* filtrati e non filtrati.

4.5.1 Identificazione di siti trascrizionalmente attivi in assenza di filtro dei dati *ChIP-on-chip*

In questa prima parte, si è voluto preliminarmente valutare la resistenza del metodo al rumore presente nei dati sperimentali, essendo questo disturbo estremamente comune in dati di tipo biomolecolare (Tab. 4.1).

Ci si è proposti, in particolare, di valutare se, in assenza di opportuni filtri applicati ai dati, è possibile ottenere risultati comparabili a quelli presentati in [Shmid 2007] variando il parametro τ, parametro dal quale dipende l'ampiezza degli intervalli genomici risultanti dall'analisi.
Come è possibile notare (Tab. 4.1), l'applicazione del metodo con un valore di τ ($\tau = 750$) evidenziante intervalli genomici di *1500* nucleotidi permette di identificare in totale *5* regioni trascrizionalmente attive, al costo di *15* falsi positivi. Una possibile interpretazione di tale risultato è che, in presenza di rumore nei dati da clusterizzare, il metodo non riesce ad identificare in maniera sufficientemente precisa il sito trascrizionale utilizzando una finestra genomica di dimensioni paragonabili a quelle di un promotore genico: se questa interpretazione fosse corretta, un'incremento del valore di τ avrebbe dovuto portare ad un aumento del numero di siti identificati (in quanto è possibile che alcuni siti siano nelle vicinanze di alcune delle regioni identificate), ma non sarebbe dovuto essere in grado di risolvere il problema dell'elevato numero di falsi positivi prodotti dall'analisi. Infatti, tale interpretazione è stata supportata dai risultati prodotti incrementando il valore di τ in modo da produrre intervalli genomici di *4000* nucleotidi. Si osservi che una opportuna variazione del parametro τ ($\tau = 2000$) ha permesso di identificare *8* siti trascrizionalmente attivi (come nei risultati presentati in [Shmid 2007]) pur non risolvendo il problema dei falsi positivi. In ogni caso, è opportuno evidenziare che tale risultato è stato ottenuto in assenza del filtro di intensità (applicato invece in [Shmid 2007]) e che le annotazioni genomiche utilizzate per il riconoscimento dei falsi positivi non sono state aggiornate: questo, di fatto, non permette di escludere che alcuni dei falsi positivi prodotti in questo esperimento non siano dovuti alla presenza di geni non noti nel 2007, periodo a cui risaliva l'ultimo aggiornamento dei dati utilizzati.

Tabella 4.1: Risultati ottenuti mediante analisi di dati non filtrati, per due valori finestra τ: $\tau = 750$ e $\tau = 2000$.

Identificazione di siti trascrizionalmente attivi			
$\tau = 750$		$\tau = 2000$	
n. cluster	positivi	n. cluster	positivi
1	1	1	1
2	1	2	1
3	1	3	2
4	1	4	2
5	2	5	3
6	2	6	3
7	1	7	3
8	1	8	3
9	1	9	4
10	2	10	5
11	2	11	5
12	3	12	6
13	3	13	6
14	4	14	7
15	4	15	7
16	4	16	7
17	4	17	7
18	4	18	7
19	5	19	8
20	5	20	8

4.5.2 Identificazione di siti trascrizionalmente attivi in presenza di filtro dei dati *ChIP-on-chip*

In questo secondo test, come precedentemente anticipato, si è applicato un filtro di intensità di valore *10* ai dati in accordo con il protocollo spe-

rimentale applicato in [Shmid 2007]. Avendo rimosso il rumore nei dati, si è deciso di testare un unico valore di τ ($\tau = 750$), evidenziante intervalli genomici compatibili con le caratteristiche biologiche dei promotori genici. Il metodo applicato è risultato essere totalmente compatibile con quello usato nel lavoro di presentazione di MADAP, ed ha reso i risultati ottenuti direttamente confrontabili (a differenza di quelli ricavati nel test precedente).

In assenza di rumore, il metodo proposto è stato in grado di rilevare correttamente ben *14* siti trascrizionalmente attivi, a differenza di MADAP che ne ha identificato solamente *8* (Tab. 4.2). Sono rimasti, comunque, *6* falsi positivi, la cui analisi dettagliata verrà presentata in un paragrafo successivo.

Le coordinate genomiche dei punti di inizio dei cluster prodotti dalla soluzione composta da *20* cluster sono le seguenti (release del genoma umano: Mar. 2006, NCBI36, hg18(UCSC)):

Human chromosome: **12**

6808487 6808787 6830441 6830841 6831641 6832741 6846908 6847702 6848302 6852597 6853297 6870157 6870428 6893788 6922844 6923403 6923718 6924118 6941817 6949542

4.5.3 Confronto dei risultati prodotti con annotazioni genomiche aggiornate

Un primo problema caratteristico delle analisi di genomica funzionale, a cui appartiene l'identificazione mediante metodi computazionali dei siti trascrizionalmente attivi, è che le annotazioni di riferimento vengono continuamente aggiornate. L'annotazione delle posizioni dei geni, in particolare, vengono aggiornate con frequenza mensile o bimestrale dalle banche dati di riferimento o, addirittura, giornalmente nel caso di banche dati specializzate (dette collettori primari), aventi lo scopo di

Tabella 4.2: Risultati ottenuti mediante analisi di dati filtrati. $\tau = 750$

Identificazione di siti trascrizionalmente attivi			
filtro	τ	n. cluster	positivi
10	750	1	1
10	750	2	2
10	750	3	3
10	750	4	3
10	750	5	3
10	750	6	3
10	750	7	4
10	750	8	5
10	750	9	6
10	750	10	7
10	750	11	7
10	750	12	8
10	750	13	9
10	750	14	9
10	750	15	10
10	750	16	11
10	750	17	12
10	750	18	13
10	750	19	13
10	750	20	14

acquisire le informazioni inerenti ai geni prodotte giornalmente nei laboratori di tutto il mondo. Appare quindi evidente come una corretta interpretazione dei risultati ottenuti non possa prescindere da un loro confronto con dati di annotazione aggiornati, soprattutto se si considera l'elevato intervallo di tempo trascorso dal periodo di esecuzione delle analisi presentate nel lavoro che descrive MADAP (avvenute nel 2007).

Tabella 4.3: Risultati ottenuti mediante analisi di dati filtrati. Comparazione con annotazione riferita alla release genomica NCBI36. Un valore 1 nella riga "Annotazioni" indica la presenza di un sito trascrizionalmente attivo nella regione genomica identificata dal cluster corrispondente riportato nella riga "Cluster".

Cluster	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Annotazioni	1	1	0	1	1	0	1	1	0	1	1	0	0	1	1	1	1	1	0	1

Un secondo problema che si deve affrontare, prima di poter effettuare un confronto dei risultati ottenuti con le annotazioni genomiche aggiornate è che, periodicamente, viene aggiornata la sequenza stessa del genoma umano. Uno degli aggiornamenti più frequenti riguarda l'inserzione di parti di genoma, relativamente piccole ma in numero rilevante. Questo causa una sorta di 'shift' ("scivolamento") delle coordinate genomiche di alcune regioni del genoma, che si concreta in una variazione delle coordinate dei geni e che impedisce, di fatto, un confronto diretto di risultati prodotti da analisi effettuare su release differenti. Ovviamente, una possibile soluzione è rappresentata dal remapping delle sequenze sulle coordinate del genoma aggiornato. Esso è realizzabile mediante allineamento delle sequenze nucleotidiche estratte dalla vecchia release del genoma alla sequenza della release corrente (Feb. 2009 GRCh37, hg19 (UCSC)). Tale approccio permette, inoltre, di verificare in maniera immediata l'overlap delle regioni identificate con annotazioni aggiornate mediante l'utilizzo di un browser genomico. Il risultato di questa caratterizzazione è illustrato in Fig. 4.6, mentre lo stato di annotazione delle regioni genomiche nella release precedente del genoma umano (NCBI36) è schematizzato in Tbl. 4.3.

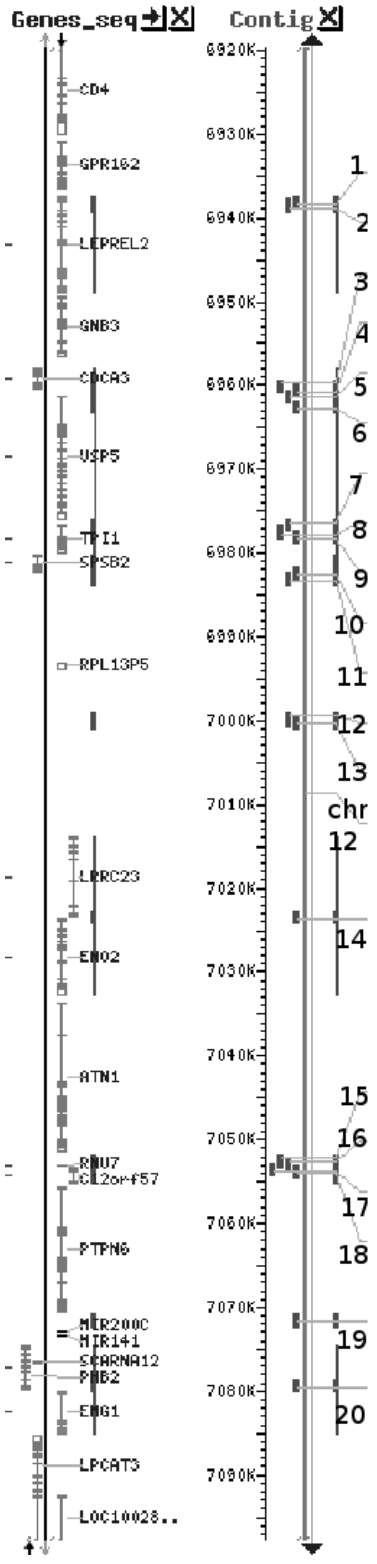

Figura 4.6: Proiezione dei risultati prodotti mediante analisi di dati filtrati sulla release corrente del genoma umano (GRCh37). I numeri a lato indicano i cluster responsabili dell'identificazione delle regioni genomiche.

4.5.4 Interpretazione dei risultati ottenuti

Nella valutazione di dati rumorosi, il metodo proposto è riuscito ad identificare un numero di siti trascrizionalmente attivi (identificati da regioni genomiche di *1500* nucleotidi) lievemente inferiore (*5* contro *8*) rispetto a quelli identificati da MADAP, pur introducendo un numero elevato di falsi positivi (*15* su *20*). L'ipotesi che tale risultato sia dovuto all'identificazione di cluster "posizionalmente" imprecisi è supportata dai risultati del medesimo test, ripetuto ampliando da *1500* a *4000* nucleotidi la dimensione degli intervalli genomici definiti sulla base dei risultati del metodo di clustering. In questo caso il metodo proposto ha identificato *8* regioni trascrizionalmente attive, producendo risultati comparabili a quelli ottenuti dal metodo MADAP operante su dati filtrati. Si può concludere quindi che il metodo proposto è in grado di rilevare siti trascrizionalmente attivi nonostante l'utilizzo di dati rumorosi, ma che l'applicazione del metodo in queste condizioni non permette di gestire efficacemente il numero dei falsi positivi prodotti ed inoltre produce risultati posizionalmente imprecisi (in termini di coordinate genomiche). Nella valutazione di dati filtrati, seguendo il medesimo approccio utilizzato durante la valutazione del metodo MADAP, il metodo proposto ha prodotto *14* cluster riconducibili a siti trascrizionalmente attivi e *6* falsi positivi utilizzando intervalli genomici di *1500* nucleotidi, e quindi compatibili con la definizione biomolecolare di promotore. Questi test sono stati effettuati utilizzando unicamente le informazioni disponibili al momento della realizzazione dei test associati alla pubblicazione del metodo MADAP. Questo secondo test ha confermato la capacità del metodo adoperato di identificare siti trascrizionalmente attivi, ed è importante notare che l'algoritmo di programmazione dinamica sviluppato produce risultati migliori sia in termini di precisione nella localizzazione delle coordinate genomiche dei siti identificati, sia in termini di numero di falsi positivi, a condizione di operare su dati opportunamente filtrati.

L'elevato numero di falsi positivi (*6* su *20* regioni identificate) richiede un'attenta analisi di queste regioni genomiche alla luce delle variazioni avvenute, negli ultimi tre anni, sia nella sequenza genomica di riferimento del genoma umano che nelle annotazioni dei geni in essa presenti. Questo test è stato realizzato proiettando i risultati ottenuti dalla valutazione dei dati dell'esperimento MADAP sulla release corrente del genoma umano ed ispezionando le annotazioni disponibili nell'ultimo aggiornamento delle annotazioni geniche (avvenuto in data 1 Ottobre 2010).

Guardando il contenuto di Fig. 4.6 si nota che i cluster prodotti identificano effettivamente siti trascrizionalmente attivi e si riesce a spiegare, almeno in parte, i *6* cluster che risultavano essere falsi positivi, se confrontati con le annotazioni disponibili al momento della pubblicazione di MADAP. In particolare (vedere Tbl. 4.3 per una lista dei falsi positivi) i cluster *3* e *6* sono nelle immediate vicinanze del punto di inizio di trascrizione del gene **CDCA3**, mentre il cluster *9* identifica una regione genomica che si trova immediatamente a valle del punto di inizio della trascrizione del gene **TPI1**. Non è possibile giustificare i falsi positivi associati ai clusters *12* e *13*, in quanto essi si posizionano in una regione genomica totalmente priva di annotazioni (allo stato attuale), mentre è degno di nota evidenziare che il cluster *19* è associato all'esistenza delle annotazioni di una coppia di microRNA (**MIR200C** e **MIR141**) non disponibili al momento della pubblicazione del metodo MADAP. È importante sottolineare quest'ultimo, inatteso risultato ottenuto durante i test, cioè l'identificazione della regione genomica contenente dei microRNA: tale tipologia di gene, scoperta relativamente di recente, è implicata nella regolazione dell'attività di altri geni e solo negli ultimi anni è divenuta oggetto di attiva e sistematica indagine da parte della comunità scientifica. Pertanto, tra i possibili sviluppi futuri dell'applicazione appena descritta, potrebbe essere di estremo interesse a livello bioinfor-

matico lo sviluppo di versioni dell'algoritmo specificamente adattate alla rilevazione di regioni genomiche codificanti microRNA.

Conclusioni

In questo lavoro è stato introdotto il 'Problema di Clustering vincolato', in cui, oltre a fissarne il numero, si sono fissate a priori anche le cardinalità dei singoli cluster. Questo problema è stato qui investigato nel caso monodimensionale, in cui gli elementi da clusterizzare erano numeri razionali; il problema è stato considerato per norme L_p con $p \geq 1$.
Preliminarmente, è stato studiato il Problema della Localizzazione del p-Centroide, classificandolo con metodi di complessità computazionale. Detta **P** la classe dei problemi risolubili in tempo polinomiale e detta **CH** la 'Counting Hierarchy' di Wagner [Wagner 1986], si è ottenuto che (**Teorema 2.1**):

1. Per ogni p razionale, il Problema della Localizzazione del p-Centroide $\in CH$;

2. Se p intero, il Problema della Localizzazione del p-Centroide $\in P$.

Si è poi mostrato che le soluzioni ottime del problema di Clustering vincolato monodimensionale verificano la cosidetta 'String Property' (**Teorema 3.1** e **Teorema 3.3**), estendendo in modo naturale al caso vincolato la medesima proprietà già provata per il problema di Clustering senza vincoli ([Edwards e Cavalli-Sforza 1965] e [Novick 2009]).
La riduzione della ricerca a soluzioni verificanti la String Property ha portato al progetto di un efficiente algoritmo per il Biclustering (**Alg BCV**); si è invece provato che il problema generale del clustering vincolato in R^1 è NP-hard (**Teorema 4.1**), e quindi esso non ammette

algoritmi risolutivi esatti in tempo polinomiale (se $\mathbf{P} \neq \mathbf{NP}$). Si è infine considerata una versione rilassata del problema di clustering vincolato, la quale ammette algoritmi risolutivi efficienti. Il primo algoritmo proposto (**mclusterdinamico**) lavora per ogni norma L_p, impiegando per p intero un tempo polinomiale. L'uso della norma L_2 ha consentito l'ottimizzazione di quest'ultimo algoritmo, portando all'algoritmo **mclusterdinamico2** che lavora in tempo $O(n^2 \cdot m)$, dove m è il numero di cluster e n la dimensione dell'istanza.

Nella parte sperimentale, sono state testate le potenzialità del nuovo algoritmo di clustering monodimensionale **mclusterdinamico2** per l'identificazione di siti trascrizionalmente attivi mediante l'analisi di dati biomolecolari (prodotti mediante analisi *ChIP-on-chip*). La qualità delle soluzioni sono state confrontate con quelle ottenute col metodo MADAP [Shmid 2007]. I risultati preliminari permettono di concludere che il metodo proposto è applicabile al problema dell'identificazione dei siti trascrizionalmente attivi.

Questa Tesi si è occupata essenzialmente del problema di Clustering vincolato monodimensionale. Il principale ambito di ricerca lasciato aperto è costituito dallo studio (risultati di analisi, algoritmi risolutivi, euristiche) del problema di Clustering vincolato su spazi d-dimensionali.

Per quanto riguarda gli algoritmi realizzati, l'algoritmo **Alg BVC** risolutivo al problema di Biclustering è polinomialmente efficiente per norme L_1 e L_2; sarebbe interessante ed utile trovare una sua implementazione efficiente anche per norme L_p, con p razionale. La NP-completezza del Problema del Clustering vincolato implica la non esistenza di algoritmi efficienti esatti: è lasciato aperto lo studio di algoritmi che diano una soluzione sub-ottima (con errore predeterminato) in tempi polinomiali.

Rispetto all'applicazione dell'algoritmo **mclusterdinamico2** all'identificazione di siti trascrizionalmente attivi, tra i futuri sviluppi è possibile annoverare lo studio dell'applicazione di diversi metodi di filtro ai dati

ChIP-on-chip in modo da valutarne sistematicamente l'impatto sulle performance del metodo: inoltre, è senz'altro possibile ed auspicabile estendere l'indagine effettuata a più ampie regioni del genoma umano.

Appendici

Appendice A

Proprietà del Centroide C_p

In questa Appendice elenchiamo e dimostriamo una serie di proprietà del Centroide C_p, particolarizzandone le proprietà in R^1. A tal fine, richiamiamo brevemente la:

Definizione di Centroide

Dato un insieme A di n vettori $x_1, x_2, ..., x_n \in R^N$, nello spazio R^N, dotato di norma $||.||_p$ $(p \geq 1)$, con '**Centroide**' (o '**Medioide**') si intende il vettore C_A tale che:

$$C_A = arg\min_{c \in R^N} \sum_{i \in A} ||x_i - c||^p$$

Supponiamo di lavorare per $N = 1$ cioè in R^1. Si hanno le seguenti proprietà:

A.1 Unicità del centroide C_p

In R^1 si ha:

$$C_p = arg\min_x \sum |x_i - x|^p$$

Lemma 1: $\forall p$ con $p \geq 1$, il Centroide C_p è unico.

Dimostrazione:

Il caso $p = 1$ è semplice in quanto C_p viene ad essere la Mediana dei numeri $(x_1, x_2, ..., x_n)$: pur non essendo teoricamente la Mediana univocamente definita, il valore C_p può essere reso unico a patto di prendere

la media aritmetica dei valori $(x_{\frac{n}{2}}, x_{\frac{n}{2}+1})$ se n pari, o il valore $x_{\lfloor \frac{n}{2} \rfloor+1}$ se n dispari.

Quindi, supponendo $p > 1$, abbiamo la funzione $y(x)$

$$y(x) = \sum_i^N |x_i - x|^p$$
$$y'(x) = p \cdot [\sum_{x_i<x}(x - x_i)^{p-1} - \sum_{x_i>x}(x_i - x)^{p-1}]$$
$$y''(x) = p(p-1) \cdot \sum_i^N |x_i - x|^{p-2}$$

se $p \neq 1$ allora $y''(x) > 0$ per ogni x e quindi $y'(x)$ è monotona crescente ed essendo $y'(x_1) < 0$ e $y'(x_N) > 0$, esiste un unico minimo C_p, tale che $y'(C_p) = 0$. Quindi, $\forall p$ con $p \geq 1$, il Centroide C_p è unico. □

A.2 Monotonicità della posizione del centroide C_p

Abbiamo il seguente:

Lemma 2: Rispetto ad una variazione delle variabili, la posizione del Centroide C_p $(\forall p > 1)$ è monotona concorde.

Dimostrazione:

Si vede subito che, $\forall j \in [1, 2, ..., n]$ e $\forall p > 1$, la posizione del centroide C_p è monotona concorde rispetto ad una variazione delle variabili. Supponiamo:

$$0 < x_1 < x_2 < .... < x_s \leq C_p \leq x_{s+1} < ... < x_n$$

si ha naturalmente, per definizione di centroide C_p:

$$\sum_{x_i>C_p}(x_i - C_p)^{p-1} - \sum_{x_i<C_p}(C_p - x_i)^{p-1} = 0$$

Se $x_j > C_p$ si ha:

$$(x_j - C_p)^{p-1} + \sum_{\substack{x_i>C_p \\ i\neq j}}(x_i - C_p)^{p-1} - \sum_{x_i<C_p}(C_p - x_i)^{p-1} = 0$$

derivando rispetto a x_j si ha:

$$(p-1)(x_j - C_p)^{p-2}(1 - \frac{\partial C_p}{\partial x_j}) + (p-1)\sum_{\substack{x_i>C_p \\ i\neq j}}(x_i - C_p)^{p-2}(-\frac{\partial C_p}{\partial x_j})$$
$$\text{- (p-1) } \sum_{x_i<C_p}(C_p - x_i)^{p-2}(\frac{\partial C_p}{\partial x_j}) = 0$$

semplificando e accorpando x_j nella sua sommatoria di origine:

$$(x_j - C_p)^{p-2} - \frac{\partial C_p}{\partial x_j}(\sum_{x_i > C_p}(x_i - C_p)^{p-2} + \sum_{x_i < C_p}(C_p - x_i)^{p-2}) = 0$$

si ha:

$$\frac{\partial C_p}{\partial x_j} = \frac{(x_j - C_p)^{p-2}}{(\sum_{x_i > C_p}(x_i - C_p)^{p-2} + \sum_{x_i < C_p}(C_p - x_i)^{p-2})}$$

In modo perfettamente analogo, se $x_j < C_p$ si ha:

$$\frac{\partial C_p}{\partial x_j} = \frac{(C_p - x_j)^{p-2}}{(\sum_{x_i > C_p}(x_i - C_p)^{p-2} + \sum_{x_i < C_p}(C_p - x_i)^{p-2})}$$

E quindi l'asserto è dimostrato. □

Appendice B

Proprietà della funzione obiettivo

Elenchiamo qui alcune utili proprietà della funzione obiettivo, richiamandone brevemente la definizione:

Definizione di Funzione obiettivo

Data una partizione $P = (B_1, B_2, ..., B_m)$ di $\{1, 2, ..., n\}$ la Funzione obiettivo F ($\Leftrightarrow$ una misura di similarità all'interno del cluster B_j), è data dalla somma degli scarti dal centroide $C_{B_j} = C_p$, cioè:

$$F(B_j) = \sum_{k \in B_j} |x_k - C_{B_j}|^p$$

dove con C_{B_j} si intende il p-centroide di B_j.

B.1 Monotonicità del valore della funzione obiettivo calcolata sul centroide C_p

Lemma 3: Il valore della funzione obiettivo calcolata sul centroide C_p è monotono concorde rispetto ad una variazione delle variabili, $\forall j \in [1, 2, ..., n]$ e $\forall p > 1$.

Dimostrazione:

Infatti, poichè si ha:

$$y = \sum_{x_i > C_p} (x_i - C_p)^p + \sum_{x_i < C_p} (C_p - x_i)^p$$

derivando rispetto a x_j, con $x_j > C_p$ si ha:

$$\frac{\partial y}{\partial x_j} = p(x_j - C_p)^{p-1}$$

la derivata è positiva e quindi, allontanadosi dal centroide, il valore della funzione aumenta, mentre derivando rispetto a x_j, con $C_p > x_j$ si ha:

$$\frac{\partial y}{\partial x_j} = -p(C_p - x_j)^{p-1}$$

cioè avvicinandosi al centroide la derivata è negativa e quindi, ad un incremento positivo dalla variabile, il valore della funzione diminuisce. □

B.2 Minimalità del valore della funzione obiettivo calcolata sul centroide C_p

Lemma 4: La funzione obiettivo calcolata sul centroide C_p ottimo è minima, cioè per un C qualsiasi il suo valore è $\geq$ rispetto al valore assunto per il C_p ottimo.

Dimostrazione:

La proprietà è evidente quando si nota che $\forall k$ e $\forall p > 1$, e che per ogni k-pla di numeri $x_1, x_2, ..., x_k$, la funzione obiettivo è convessa rispetto al centroide C_p, e quindi esiste un unico centroide C_p ottimo sul quale il valore assunto dalla funzione obiettivo è minimo. □

Sia $p > 1$ e C_A il centroide ottimo:

$$C_A = arg\,\min_{c \in \mathbf{R}} \sum_{i \in A} |x_i - c|^p$$

Allora:

Teorema: Se $G(\mu) = \sum_{i \in A} |x_i - \mu|^p$ allora $G(\mu)$ è convessa, cioè

$G(\alpha \cdot \mu_1 + \beta \cdot \mu_2) \leq \alpha \cdot G(\mu_1) + \beta \cdot G(\mu_2)$, con $\alpha \geq 0, \beta \geq 0$, $\alpha + \beta = 1$

Dimostrazione:

$G(\mu) = \sum_{i \in A} |x_i - \mu|^p$, allora

$G(\alpha \cdot \mu_1 + \beta \cdot \mu_2) = \sum_{i \in A} |x_i - (\alpha \cdot \mu_1 + \beta \cdot \mu_2)|^p = \sum_{i \in A} |(\alpha + \beta) \cdot x_i - (\alpha \cdot \mu_1 + \beta \cdot$

$\mu_2)|^p = \sum_{i\in A} |(\alpha\cdot(x_i-\mu_1)+\beta\cdot(x_i-\mu_2)|^p \leq$ (per le proprietà della metrica)
$\leq \sum_{i\in A}(\alpha\cdot|x_i-\mu_1|+\beta\cdot|x_i-\mu_2)|)^p \leq \sum_{i\in A}(\alpha\cdot|x_i-\mu_1|^p+\beta\cdot|x_i-\mu_2|^p) =$
$\alpha\cdot\sum_{i\in A}|x_i-\mu_1|^p+\beta\cdot\sum_{i\in A}|x_i-\mu_2|^p = \alpha\cdot G(\mu_1)+\beta\cdot G(\mu_2)$.
Ne segue che, in generale, l'insieme che minimizza $G(\mu)$ è un convesso e quindi C_A è il punto dove la funzione obiettivo assume il valore minimo.
□

Appendice C

Metodo di approssimazione di Newton

Questo metodo permette di approssimare le soluzioni di un'equazione $f(x) = 0$, cioè di calcolare gli zeri della funzione $y = f(x)$ in un intervallo $[a, b]$ a patto di saper calcolare la funzione e la sua derivata in tutto l'intervallo.

In primo luogo, è necessario separare le soluzioni e cioè individuare gli intervalli nei quali cadono le singole soluzioni, in quanto esso si applica dopo avere determinato un intervallo che contiene una sola radice.

Il metodo consiste nel sostituire alla curva $y = f(x)$ la tangente in uno dei due punti che hanno come ascissa gli estremi dell'intervallo $[a, b]$ e assumere, come valore approssimato della radice, l'ascissa x_t del punto (punto di innesco dell'algoritmo) in cui la tangente interseca l'asse delle x internamente all'intervallo. Supponiamo che nell'intervallo $[a, b]$ la funzione e le sue derivate prima e seconda esistano e siano continue e che la derivata prima e seconda siano diverse da zero, in modo che la concavità non cambi di segno nell'intervallo. Conviene tracciare la tangente nell'estremo dell'intervallo in cui la funzione e la sua derivata seconda hanno lo stesso segno. Utilizzando l'equazione della retta generica (o fascio di rette) per a:

$$y - f(a) = m(x - a)$$

sostituendo m con $f'(a)$ e imponendo $y = 0$ (cioè l'attraversamento dell'asse delle ascisse) si ha:

$$x_0 = a - \frac{f(a)}{f'(a)}$$

Il procedimento si può iterare, in quanto abbiamo determinato il nuovo intervallo $[x_0, b]$ contenente la radice cercata e, ripetendo il procedimento visto, otteniamo una nuova approssimazione della radice (intersezione della seconda tangente con l'asse delle x):

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

In generale, chiamando iterativamente x_n l'ennesima approssimazione e x_{n+1} quella successiva, si ha la classica formula di ricorrenza di Newton:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Il procedimento è chiaramente convergente, cioè fissato un errore ϵ piccolo a piacere, si troverà sempre una approssimazione x_n della radice cercata la cui distanza da essa, in valore assoluto, è minore di tale ϵ: inoltre, con le ipotesi fatte, si dimostra che la successione delle x_n converge alla radice piuttosto rapidamente. In tal senso, vale il seguente (di cui si omette la dimostrazione):

Teorema: Siano $f \in C^2(I)$, I = intorno della radice α, $f(\alpha) \neq 0$ e $x_0 \in I$, allora:

$$\lim_{n\to\infty} \frac{\alpha - x_{n+1}}{(\alpha - x_n)^2} = -\frac{f''(\alpha)}{2 \cdot f'(\alpha)}$$

In altri termini la convergenza, benchè locale (cioè non valida $\forall$ I), è quadratica, mentre in caso di radice multipla ($\Leftrightarrow f'(\alpha) = 0$) la convergenza è più lenta, precisamente lineare.

Bibliografia

[ABKM 2006] Allender E., Burgisser P., Kjeldgaard-Pedersen J. 1987. On the complexity of numerical analysis. *In Proc. 21st Ann. IEEE* Conf. 2006.

[Allender 2009] Allender E., Brgisser P., Kjeldgaard-Pedersen J., Miltersen P. B. 2009. On the Complexity of Numerical Analysis. *SIAM J. Comput.* 38(5): 1987-2006.

[Aloise 2009] Aloise D., Deshpande A., Hansen P., Popat P. 2009. *NP-hardness of Euclidean sum-of-squares clustering.* Machine learning , 245-248.

[Anderberg 1973] Anderberg M. R. 1973. *Cluster Analysis for Applications.* Academic Press, Inc., New York, NY.

[ASBR 2008] Abeel T., Saeys Y., Bonnet E., Rouzo P., Van de Peer Y. 2008. *Generic eukaryotic core promoter prediction using structural features of DNA.* Genome Research 18, 310-23.

[Ball e Hall 1965] Ball G. H. Hall D. J. 1965. *ISODATA, a novel method of data analysis and classification.* Tech. Rep.. Stanford University, Stanford, CA.

[Baraldi e Alpaydin 2002] Baraldi , A. and Alpaydin , E. 2002. Constructive feedforward ART clustering networks Part I and II . *IEEE Transactions on Neural Networks* , 13 (3): 645-677.

[Ben-David 2006] Ben-David S., Von Luxburg U., Pl D. 2006. *A Sober Look at Clustering Stability.* COLT 2006: 5-19.

[Ben-David 2008] Ben-David S., Von Luxburg U. 2008. *Relating Clustering Stability to Properties of Cluster Boundaries.* COLT 2008: 379-390.

[Ben-David 2009] Ackerman M., Ben-David S. 2009. *Which Data Sets are Clusterable? A Theoretical Study of Clusterability* Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics.

[Bernasconi Codenotti 1998] Bernasconi A., Codenotti B. 1998. *Introduzione alla Complessità Computazionale*, Springer-Verlag Italia, Milano.

[Bongartz Calamai 1994] Bongartz I., Calamai P.H.,Conn A.R., 1994. *A projection method for L_p norm location-allocation problems*, Mathematical Programming 66 283-312.

[Canny 1988] Canny, J., 1988. *The complexity of robot motion planning*, *ACM Doctoral Dissertation Awards*; Vol. 1987, MIT Press, Cambridge, MA, USA.

[Cheng 1995] Cheng C. H. 1995. A branch-and-bound clustering algorithm. *IEEE Trans. Syst. Man Cybern. 25*, 895-898.

[Claude 2008] Claude F., Navarro G. 2008. *Indexing Straight-Line Programs* - CiteSeerX Scientific Literature Digital Library and Search Engine, United States.

[Cook 1971] Cook, S. A. 1971. *The complexity of theorem-proving procedures.* STOC '71: Proceedings of the third annual ACM symposium on Theory of computing. Shaker Heights, Ohio, United States. New York, NY, USA; 151-158.

[Dempster 1977] Dempster A. P., Laird N. M., Rubin D.B. 1977. Maximum likelihood from incomplete data via the EM algorithm. *J. Royal Stat. Soc. B. 39*, 1, 1-38.

[Diday 1973] Diday E. 1973. The dynamic cluster method in non-hierarchical clustering. *J. Comput. Inf. Sci.* 2, 61-88.

[Drineas 2004] Drineas P., Frieze A., Kannan R., Vempala S., Vinay V. 2004. *Clustering Large Graphs via the Singular Value Decomposition.* Machine Learning, 56, 9-33.

[Dubes 1987] Dubes R. C. 1987. How many clusters are best? An experiment. *Pattern Recogn. 20,* 6 (Nov. 1, 1987), 645-663.

[Duda et al. 2001] R.O. Duda, P.E. Hart e D.G. Stork, 2001. *Pattern Classification*, 2 nd edition. New York, NY: John Wiley Sons.

[Edwards e Cavalli-Sforza 1965] Edwards A.W.F., Cavalli-Sforza L.L., 1965. A method for cluster analysis, *Biometrics* 21, 362-375.

[Fisher 1958] Fisher W. D. 1958. On Grouping for Maximum Homogeneity, *Journal of the American Statistical Association, 53*, 789-798.

[Fogel et al. 1965] Fogel L. J., Owens A. J., Walsh M. J. 1965. *Artificial Intelligence Through Simulated Evolution.* John Wiley and Sons, Inc., New York, NY.

[Garey Johnson 1975] Garey M.R., Johnson D.S. 1975. Complexity results for multiprocessor scheduling under resource constraints. *SIAM Journal on Computing 4*, 397-411.

[Garey Graham Johnson 1976] Garey M.R., Graham R.L., Johnson D.S. 1976. Some NP-Complete Geometric problems. *Annual ACM Symposium on Theory of Computing archive*, 10-22.

[Garey Johnson 1979] Garey M.R., Johnson D.S. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman, San Francisco.

[Gasieniec 2003] Gasieniec L., Potapov I. 2003. *Time/space efficient compressed pattern matching.* Fund. Inf., 56 (1-2), 137-154.

[Gill 1977] Gill J., 1977. *Computational complexity of probabilistic Turing machines.* SIAM Journal on Computing, 6 (4), 675-695.

[Goldberg 1989] Goldberg D. E. 1989. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley Publishing Co., Inc., Redwood City, CA.

[Grefenstette 1986] Grefenstette J. 1986. Optimization of control parameters for genetic algorithms. *IEEE Trans. Syst. Man Cybern. SMC-16*, 1 (Jan./Feb. 1986), 122-128.

[Grower 1967] Grower J.C. 1967. A comparison of some Methods of Cluster Analysis *Biometrics, 23*, 623-637.

[Hansen 1998] Hansen P., B. Jaumard, N. Mladenovic, 1998. *Minimum Sum of Squares Clustering in a Low Dimensional Space*, Journal of Classification 15(1), 37-55.

[Harding 1967] Harding E.F. 1967. The numbers of partitions of a set of N in k dimensions induced by Hyperplanes, *Proceedings of the Edinburgh Mathematical Society (Series II), 15*, 285-289.

[Hertz et al. 1991] Hertz J., Krogh A., Palmer R. G. 1991. *Introduction to the Theory of Neural Computation.* Santa Fe Institute Studies in the Sciences of Complexity lecture notes. Addison-Wesley Longman Publ. Co., Inc., Reading, MA.

[Holland 1975] Holland J. H. 1975. *Adaption in Natural and Artificial Systems.* University of Michigan Press, Ann Arbor, MI.

[Huffman 1952] Huffman D.A., 1952. *A Method for the Construction of Minimum-Redundancy Codes*, Proceedings of the I.R.E., September 1952, 1098-1102.

[Jain e Dubes 1988] A. K. Jain, R. C. Dubes, 1988. *Algorithms for Clustering Data*. Prentice-Hall advanced reference series. Prentice-Hall, Inc., Upper Saddle River, NJ.

[Jain e Mao 1994] Jain A. K. Mao J. 1994. Neural networks and pattern recognition. In *Computational Intelligence: Imitating Life*, J. M. Zurada, R.J. Marks, and C. J. Robinson, Eds. 194-212.

[Jain e Mao 1996] Jain A. K. Mao J. 1996. Artificial neural networks: A tutorial. *IEEE Computer 29* (Mar.), 31-44.

[Jain Murty Flynn 1999] Jain A. K., Murty M. N., Flynn P. J. 1999. Data clustering: A review. *ACM Computing Surveys*, Vol 31, No 3 September 1999.

[Jolliffe 2002] Jolliffe I.T. 2002. Principal Component Analysis, Series: *Springer Series in Statistics*, 2nd ed., Springer, NY, XXIX, 487 p. 28 illus.

[Jones e Beltramo 1991] Jones D., Beltramo M. A. 1991. Solving partitioning problems with genetic algorithms. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, 442-449.

[Kim 2005] Kim T.H. et al. 2005. *A high-resolution map of active promoters in the human genome.* Nature, 436: 876-880.

[King 1967] King B. 1967. Step-wise clustering procedures. *J. Am. Stat. Assoc. 69*, 86-101.

[Klein e Dubes 1989] Klein R. W. Dubes R. C. 1989. Experiments in projection and clustering by simulated annealing. *Pattern Recogn. 22*, 213-220.

[Kohonen 1989] Kohonen T. 1989. Self-Organization and Associative Memory. 3rd ed. *Springer information sciences series.* Springer-Verlag, New York, NY.

[Kolmogorov 1968] Kolmogorov A. N. 1968. Three approaches to the quantitative definition of information. *Internat. J. Comput. Math.*, 2: 157-168.

[Koontz et al. 1975] Koontz W. L. G., Fukunage K., Narendra P. M. 1975. A branch and bound clustering algorithm. *IEEE Trans. Comput. 23*, 908-914.

[Lempel Ziv 1977] Ziv J. Lempel A. 1977. *A Universal Algorithm for Sequential Data Compression*, IEEE Transactions on Information Theory, 23(3), pp. 337-343, May 1977.

[Lu e Fu 1978] Lu S.Y., Fu K.S. 1978. A Sentence-to-Sentence clustering procedure for pattern analysis. *IEEE Trans. Syst. Man Cybern. 8,* 381-389.

[MacQueen 1967] MacQueen J. 1967. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, 281-297.

[Mao e Jain 1995] Mao J., Jain A. K. 1995. Artificial neural networks for feature extraction and multivariate data projection. *IEEE Trans. Neural Netw. 6*, 296-317.

[Mitchell 1997] Mitchell T. 1997. *Machine Learning.* McGraw-Hill, Inc., New York, NY.

[MRS 2007] Manning C. D., Raghavan P., Schtze H. 2007. *Introduction to Information Retrieval.* Cambridge University Press.

[Nagy 1968] Nagy G. 1968. State of the art in pattern recognition. *Proc. IEEE 56*, 836-862.

[Novick 2009] Novick B. 2009. Norm statistics and the complexity of clustering problems *Discrete Applied Mathematics*, v.157 n.8, 1831-1839.

[Oja 1982] Oja E. 1982. A simplified neuron model as a principal component analyzer. *Bull. Math. Bio. 15*, 267-273.

[Papadimitriou 1993] Papadimitriou C. 1993. *Computational Complexity* (1st edition ed.). Addison Wesley. ISBN 0-201-53082-1. Chapter 19: Polynomial space, 455-490.

[Pearson 1901] Pearson, K. 1901. *On Lines and Planes of Closest Fit to Systems of Points in Space*. Philosophical Magazine 2 (6): 559-572.

[Rao 1971] Rao M.R. 1971. Cluster analysis and Mathematical programming, *Journal of the American Statistical Association, 66*, 622-626.

[Rose et al. 1993] Rose K., Gurewitz E., Fox G. C. 1993. Deterministic annealing approach to constrained clustering. *IEEE Trans. Pattern Anal. Mach. Intell. 15*, 785-794.

[Ruspini 1969] Ruspini E. H. 1969. A new approach to clustering. *Inf. Control 15*, 22-32.

[Savitch 1970] Savitch W. J. 1970. *Relationships Between Nondeterministic and Deterministic Tape Complexities*, Journal of Computer and System Sciences, 4, 177-192.

[Schwefel 1981] Schwefel H. P. 1981. *Numerical Optimization of Computer Models.* John Wiley and Sons, Inc., New York, NY.

[Scott e Symons 1971] Scott A.J., Symons 1971. On the Edwards and Cavalli-Sforza Method of Cluster Analysis, *Biometrics, 27*, 217-219.

[Selim e Al-Sultan 1991] Selim S. Z., Al-Sultan K. 1991. *A simulated annealing algorithm for the clustering problem.* Pattern Recogn. 24, 10 (1991), 1003-1008.

[Sethi e Jain 1991] Sethi I. Jain A. K., Eds. 1991. *Artificial Neural Networks and Pattern Recognition: Old and New Connections.* Elsevier Science Inc., New York, NY.

[Shmid 2007] Schmid C. D., Sengstag T., Bucher P., Delorenzi M. 2007. *MADAP, a flexible clustering tool for the interpretation of one-dimensional genome annotation data.* Nucleic Acids Res, 35, W201-205.

[Sneath e Sokal 1973] Sneath P. H. A. Sokal, R. R. 1973. *Numerical Taxonomy.* Freeman, London, UK.

[Symon 1977] Symon M. J. 1977. Clustering criterion and multi-variate normal mixture. *Biometrics 77*, 35-43.

[Vijay 2002] Vijay K. Rohatgi, A. K. Md. Ehsanes Saleh. 2002. *An introduction to Probability and Statistics*, 2nd edition, J.Wiley.

[Vinod 1969] Vinod H.D., 1969. *Integer programming and the theory of grouping.* Journal of the American Statistical Association 64.

[Wagner 1986] Klaus W. Wagner 1986. *The complexity of combinatorial problems with succinct input representation.* Journal Acta Informatica 23(3), 325-356.

[Zadeh 1965] Zadeh L. A. 1965. Fuzzy sets. *Inf. Control 8*, 338-353.