POLITECNICO DI MILANO
Corso di Laurea Magistrale in Ingegneria Informatica
Dipartimento di Elettronica, Informazione e Bioingegneria

Progetto e realizzazione di un framework per Neosperience sul clustering di reti sociali

Relatore: Prof. Pier Luca Lanzi
Correlatore: Ing. Luca Bianchi

Tesi di Laurea di:
Nicola PADOVANO, matricola 746059
Elia POLO, matricola 760649

Anno Accademico 2013-2014

Sommario

Questo lavoro di tesi fornisce un insieme di strumenti e metodologie di supporto alla segmentazione di mercato, cioè a quel processo di marketing che suddivide un variegato insieme di clienti in gruppi omogenei (o *segmenti*) caratterizzati da simili comportamenti d'acquisto. Con l'avvento dei Big Data in generale e la pervasività dei Social Network in particolare, si è assistito ad una esplosione della complessità delle basi di dati aziendali, che possono essere analizzate efficacemente solo con l'ausilio di moderne tecniche mutuate dalla cluster analysis. Nei prossimi capitoli descriveremo nel dettaglio lo stato dell'arte della analisi dei dati, individuando gli strumenti di ultima generazione che sono stati indispensabili per raggiungere gli obiettivi del lavoro: dalla raccolta, analisi e preparazione dei dati fino all'individuazione di innovativi algoritmi di clustering che operano sia su dataset classici che su complessi grafi con attributi; successivamente mostreremo come abbiamo implementato, integrato ed utilizzato ognuna di queste tecniche, evidenziandone gli aspetti peculiari e i punti di forza e di debolezza. Infine, presenteremo delle linee guida per suggerire le migliori parametrizzazioni dei vari algoritmi ed elevare, di conseguenza, la qualità dei risultati.

Abstract

This thesis provides a set of tools and methodologies to support market segmentation, that is the marketing process that divides a diverse set of customers into homogeneous groups (or segments) characterized by similar buying behavior. With the advent of Big Data in general and the pervasiveness of social networks in particular, there has been an explosion of the complexity of corporate databases, which can be effectively analyzed only with the help of modern techniques borrowed from the field of cluster analysis. This work describes in detail the state of the art of data analysis, identifying the tools of last generation that have been essential in achieving the main goals: from the collection, study and preparation of the data up to the identification of innovative clustering algorithms that operate on both classic datasets and complex attributed graphs; then, we will show how each of these techniques was implemented, integrated and used, highlighting its peculiar aspects, strengths and weaknesses. Finally, we will present the guidelines to suggest the best parameter settings of the various algorithms and thus enhance the quality of the clustering results.

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

L'evoluzione digitale degli ultimi anni ha portato alla nascita e diffusione di strumenti di comunicazione centrati sull'utente, producendo uno stravolgimento del paradigma classico di marketing. Le aziende devono quindi fronteggiare un cambiamento radicale nel modo di presentare i propri prodotti, che vede innovati i contenuti, i canali e la prospettiva del messaggio pubblicitario. Per contro, si propone alle aziende la possibilità di estrarre informazioni inedite (e quindi valore) dagli attuali strumenti di connessione tra gli individui: i social network, tra tutti, racchiudono per ciascun utente le relazioni sociali e il *sentiment* (le opinioni, gli interessi e l'affiliazione a idee, prodotti, marchi, individui). Si apre quindi la possibilità e l'esigenza di estrarre ed elaborare tali dati, utilizzandoli per conseguire con maggiore efficacia l'*engagement*, ossia il primo contatto con gli utenti e la loro conversione in clienti. A tal fine, è essenziale identificare i gusti e gli interessi di ciascun individuo, e proporgli inserzioni personalizzate che rispondano o addirittura anticipino una sua reale esigenza. Ciò ha portato a ridefinire l'idea stessa della segmentazione di mercato—la disaggregazione dei clienti in gruppi omogenei per esigenze e comportamento di acquisto—per cui la ripartizione in aree geografiche, demografiche e psicografiche è obsoleta o insufficiente. Queste realtà, la necessità di profilare gli utenti e il diluvio di informazione grezza che ciascuno di noi divulga su internet, sono fortemente sinergiche, ma richiedono una laboriosa raffinazione per poter essere impiegate nel marketing.

La segmentazione di mercato può essere costruita a partire dalle divisioni identificate mediante tecniche di clustering, un processo che organizza una popolazione in gruppi formati da individui affini. Un esperto dovrà poi attribuire un significato (profiling) a tali gruppi, comporli per caratterizzare una tipologia di consumatori tramite le variabili usate per il clustering nonché le

caratteristiche demografiche, geografiche e comportamentali degli individui, al fine di elaborare una strategia di marketing peculiare per ciascun segmento.
Pertanto, per avvalersi dell'informazione proveniente dai social network a fini commerciali, è necessario definire una metodologia per il clustering di grandi volumi di dati: profili utente con un elevato numero di attributi e talvolta arricchiti da una tela di relazioni sociali. Questo lavoro è stato motivato da Neosperience, una azienda che sviluppa strumenti software per la *Digital Customer Experience*. In particolare, il modulo *Right Time Personalization* (RTP) permette di aggregare e selezionare i prodotti di un marchio in maniera personalizzata per un ciascun cliente, a partire da metriche di rilevanza definite dal Brand e basate sui dati del profilo social dell'utente stesso. Nell'ambito dello sviluppo ed evoluzione di tale prodotto, Neosperience ha manifestato l'esigenza di fornire ai propri clienti un framework per il clustering dei dati estratti, proponendo quindi uno strumento di supporto al processo di segmentazione. L'obiettivo di questo lavoro di tesi è stato la realizzazione di tale framework, con particolare attenzioni alle problematiche connesse alla elaborazione dei dati, oggi comunemente noto come Big Data. Il lavoro è iniziato dall'analisi degli algoritmi esistenti in letteratura che potessero adattarsi al tipo, dimensione e volume dei dati su cui avremmo operato. In seguito sono stati raccolti i dati, una ampia collezione di profili Facebook, composti da informazioni di profilo, preferenze (*like*) e connessioni sociali (*friend*). Una volta ottenuto il codice degli algoritmi, i dati sono stati ripuliti e preparati per le tecniche di clustering selezionate. Dall'esecuzione degli algoritmi su svariati campioni del dataset completo abbiamo potuto evidenziare i punti di forza e debolezza di ciascun approccio. Tramite la misurazione di indici di qualità, siamo in grado di suggerire qual è la miglior parametrizzazione di ciascun algoritmo su un dataset in input e di svolgere una analisi comparativa sulle prestazioni delle diverse tecniche.

1.1 Contesto del lavoro

Il clustering è un processo che, partendo da una definizione di similarità e una popolazione, produce una suddivisione degli individui in gruppi o *cluster*, al fine di massimizzare la similarità intra-cluster, ovvero l'omogeneità degli elementi raggruppati insieme, e la dissimilarità inter-cluster, ossia accentuare le differenze tra gruppi diversi. Si tratta di un procedimento *data-driven*, inevitabilmente determinato dai dati e dalle esigenze del committente; *clustering is in the eye of the beholder*, e non ha pertanto senso dettare quale algoritmo e scelta dei parametri siano sistematicamente ottimali. Nel caso

in esame, tuttavia, la struttura dei dati è stabile e ciò permette di avanzare delle linee guida per selezionare preliminarmente un ventaglio di approcci di maggior successo.
Negli ultimi anni, lo sviluppo delle capacità di memorizzazione ed elaborazione dei dati, unitamente alla diffusione pervasiva di Internet, hanno prodotto una svolta nell'era digitale, nella quantità e qualità di informazione nascosta nei dati che una tecnica di clustering potrebbe svelare. Questa rivoluzione ha preso il nome di *Big Data*. Big Data indica la disponibilità di enormi masse di dati, strutturati e non strutturati: una miniera di informazione grezza che richiede tecniche innovative di elaborazione per capire profondamente la realtà in cui si opera e prendere decisioni migliori[1]. Il paradigma dei Big Data è articolato in tre V: Volume Velocità Varietà.

Volume Al crescere del volume dei dati, nella valutazione di un algoritmo assumono rilievo non soltanto la qualità dei suoi risultati, ma anche la complessità spaziale e temporale. Inoltre, quando si parla di volume dei dati non ci si riferisce unicamente alla loro cardinalità, ma anche al numero di proprietà o dimensioni associate ad ogni individuo. L'effetto della *dimensionalità* elevata sul clustering è duplice: da un lato, le dimensioni irrilevanti, quegli attributi rispetto ai quali non c'è aggregazione, costituiscono rumore per gli algoritmi; dall'altro, alcune definizioni di distanza (ad esempio quella euclidea) non riflettono più la reale entità delle differenze tra gli individui [3, 14].

Velocità e Variabilità Una rete sociale è un oggetto dinamico, nella struttura e nei contenuti. Ogni giorno si sviluppano nuove connessioni, tanto tra individui—*friend* in Facebook e *follower* in Twitter—quanto verso entità—*like* in Facebook e *checkin* in Foursquare. Anche la velocità con cui la nostra impronta digitale evolve varia a seconda dei singoli dati che consideriamo: se da un lato il luogo o la data di nascita sono permanenti, al contrario le amicizie, le relazioni sentimentali e i gusti sono via via più volubili, e devono essere raccolti ed analizzati prima che diventino obsoleti.

Varietà I dati si manifestano in innumerevoli forme: pur avendo fissate le sorgenti di informazione—Facebook, Twitter e Foursquare—da esse si ricavano tabelle strutturate, flussi di messaggi ed ogni sorta di conte-

[1] "Big data is high volume, high velocity, and/or high variety information assets that demand cost-effective, innovative forms of information processing to enable enhanced decision making, insight discovery and process optimization" - Gartner [58]

nuto multimediale. La sfida è trovare forme adeguate di organizzazione dei dati e tecniche di analisi capaci di adattarsi alla varietà.

Alcune proprietà notevoli delle reti sociali contribuiscono tuttavia a dominare la variabilità nei dati. In una rete sociale ogni nodo è densamente connesso ai propri vicini, cioè esibisce un intrinseco clustering locale; per esempio, è frequente che i nostri amici siano anche amici tra loro. Inoltre, la distanza media tra una coppia arbitraria di nodi di una rete sociale è sorprendentemente bassa rispetto alla dimensione del grafo[2]; tale proprietà è nota come *teoria del mondo piccolo*. Da ultimo, la somiglianza genera connessioni; questo principio, l'*omofilia*, spiega perché siamo spesso simili ai nostri amici, per età, esperienze passate e passioni. In conclusione, la varietà dei dati provenienti da reti sociali non è libera, ma esibisce delle regolarità, specialmente nella cerchia ristretta di ciascun individuo.
Prima di poter essere sottoposti al clustering, i dati grezzi devono tuttavia essere elaborati per elevarne la qualità. Questa fase preparatoria, detta *preprocessing*, si sostanzia dei seguenti passi: pulizia, integrazione, riduzione e trasformazione dei dati. La *pulizia* dei dati ha lo scopo di identificare le incongruenze, completare i dati mancanti, attenuare il rumore e rimuovere le anomalie. Dall'*integrazione*, le differenti sorgenti di dati confluiscono in un unico archivio, avendo cura di definire un formato unico, coerente e privo di ridondanza verso il quale convertire le sorgenti. Partendo da questo modello dei dati è possibile definire le caratteristiche dell'algoritmo di clustering ideale, che farà da guida nella nella selezione delle metodologie esistenti in letteratura. Poiché tecniche diverse richiedono specifiche proprietà degli input per offrire i migliori risultati, i due passi successivi sono necessari per adattare i dati all'algoritmo scelto. La *riduzione* consiste nel comprimere la forma dei dati, in particolare il numero di attributi o il numero di individui, cercando di preservarne intatta la sostanza, l'informazione nascosta. In ultimo si esegue la *trasformazione*, che agisce sulla scala, sul tipo e sulla granularità dei singoli attributi. Oltre ai requisiti imposti dai dati, una caratteristica desiderabile per un algoritmo applicato alle reti sociali è la capacità di individuare comunità parzialmente sovrapposte o addirittura annidate, nonché identificare cluster omogenei per diversi sottoinsiemi di attributi (*subspace clustering*). Infine, l'algoritmo dovrebbe essere scalabile nelle dimensioni e nella cardinalità dei dati.
Durante l'analisi sperimentale, si porrà il problema di decidere quale scelta dei parametri dell'algoritmo—ad esempio il numero di comunità da indi-

[2]Una rete è detta *small-world network* se la distanza tra due individui scelti a caso è proporzionale a $logN$, dove N è il numero di nodi

viduare nella popolazione—produca il miglior risultato. Sebbene esistano svariati indici numerici di qualità, le proprietà desiderabili di un clustering sono la purezza e la sintesi. La purezza misura quanto ciascun gruppo è composto da individui simili, privo di elementi estranei che appartengono ad altri gruppi o sono atipici rispetto all'intera popolazione. La sintesi giudica quanta inutile complessità e ridondanza vi sia nel clustering, ed indirettamente la capacità del modello di essere generalizzato per descrivere il fenomeno da cui i dati sono estratti.

1.2 Breve descrizione del lavoro

L'esigenza principale di Neosperience è quella di disporre di uno strumento, il più possibile automatico, che possa valorizzare i dati grezzi dei propri clienti. Questi infatti possiedono dei grandi database di *clienti finali* che, di per sé, non presentano nessun valore addizionale in quanto mancano di una semantica associata ad essi. Una delle aree più importanti del marketing è la *segmentazione di mercato*, che si prefigge lo scopo di trovare regolarità (di esigenze e comportamenti d'acquisto) all'interno di un numeroso insieme di individui. È stato proprio questo l'obiettivo di Neosperience: dare un significato ai dati tratteggiando le diverse tipologie di clienti che ne emergono. Da un lato, la segmentazione è considerata un compito creativo basato sull'intuizione del dirigente aziendale nell'individuare aree potenziali di mercato, dall'altro necessita, sicuramente, di una metodologia precisa e scrupolosa che consente di determinare almeno i confini teorici entro i quali poter prendere delle decisioni ponderate. La cluster analysis viene in aiuto in questo senso, mettendo a disposizioni metodi automatici per identificare gruppi omogenei di individui secondo una definizione di similarità tra questi. Delineiamo ora quali sono stati i passi fondamentali del lavoro di tesi.

Analisi della letteratura

Come in ogni campo della scienza, per un determinato problema non esiste un'unica soluzione ma sempre è possibile scegliere o progettare una vasta gamma di strumenti che propongono dei risultati diversi e con una qualità che varia a seconda delle loro caratteristiche. Ciò porta a vantaggi immediati: a causa della varietà delle basi di dati e della loro mutevolezza nel tempo, è necessario affidarsi ad un ventaglio di soluzioni in modo tale da poter scegliere quella che, per un certo insieme di individui in ingresso, risulti essere ottimale. Di conseguenza, l'analisi della letteratura non è stata focalizzata sulla ricerca di un solo algoritmo capace di risolvere qualsiasi problema; si è

cercato, invece, di avvalersi di un insieme di algoritmi in cui ognuno potesse intervenire sui dati in input a seconda dei propri meriti: come, ad esempio, la capacità di gestire grafi completi o con componenti isolate, di affrontare dataset con un elevato numero di attributi o con valori mancanti, oppure di essere scalabile in funzione della dimensione dell'input e delle performance dell'hardware.
Infine, oltre agli algoritmi di clustering, sono state studiate metodologie di *data preprocessing* e pacchetti software (R[3], Matlab[4], Gremlin[5]) per la raccolta e l'analisi di dati.

Raccolta e analisi dati

Col fine di ottenere una base di dati su cui eseguire e valutare gli algoritmi selezionati abbiamo sviluppato un'applicazione PHP che recupera il profilo Facebook di un utente e dei suoi amici. In particolare, per ogni utente dell'applicazione, si raccolgono le informazioni di profilo (genere, città natale, città di residenza, istruzione), le preferenze (le pagine su cui si è cliccato *like*), i legami d'amicizia e, di nuovo, informazioni di profilo e preferenze degli amici. In questa maniera, da ogni utente si ricava una rete personale (*ego-network*) che, composta con le altre, ha permesso la creazione di un unico grafo con attributi.
Il primo passo è stato quello di ripulire il grafo da utenti con un'alta percentuale di informazioni mancanti (sia relazionali che di profilo); successivamente si è svolta, tramite considerazioni statistiche, l'imputazione[6] dei *missing value* e la selezione di attributi rilevanti per disporre di informazioni complete e significative su ogni utente. Infine, è stato sviluppato un algoritmo di campionamento che, da un lato, riuscisse a creare sottografi significativi dal punto di vista topologico e degli attributi, dall'altro, che presentasse una forte componente casuale per modellare la varietà dei dati in input che potrebbero essere usati in seguito. Conclusa questa fase abbiamo quindi ottenuto un insieme di sottografi su cui è possibile eseguire e valutare gli algoritmi di clustering. Infine, ogni sottografo è stato etichettato con una serie di misurazioni topologiche che lo "identificano".

[3] http://www.r-project.org/
[4] http://www.mathworks.it/
[5] https://github.com/tinkerpop/gremlin/wiki
[6] Il termine imputazione indica una serie di tecniche aventi l'obiettivo di prevedere o "stimare" i valori sostitutivi dei dati mancanti

Valutazione e confronto degli algoritmi

Ogni algoritmo presenta una serie di parametri iniziali che devono essere impostati accuratamente per elevare la qualità dei risultati: è quasi sempre possibile specificare, ad esempio, il numero di cluster da identificare, oppure il numero massimo di attributi che caratterizzeranno un cluster. Tramite la misurazione dei tempi d'esecuzione e degli *indici di validità*—stime che valutano quanto sia appropriato un certo clustering in relazione alle caratteristiche degli individui—si è raggiunto l'obiettivo "zero" del lavoro: capire qual è la miglior parametrizzazione di un algoritmo su un dataset in input. La fase successiva è stata quella di confrontare e combinare questi risultati per ogni algoritmo a disposizione: in questo modo abbiamo raccolto tutte le informazioni necessarie per condurre delle analisi più complesse. Fissati i dati sotto esame è possibile, infatti, rispondere a domande come le seguenti:

- Come varia la stessa prestazione P (ad esempio il tempo d'esecuzione) di due algoritmi al variare dei loro parametri? È possibile decidere quale dei due algoritmi è mediamente il migliore in relazione a P?
- Valutando l'andamento dei vari indici prestazionali, esiste un trend che permette di decidere qual è l'algoritmo, in generale, più efficace?

Infine, abbiamo sviluppato un tool per attribuire un significato "descrittivo" al risultato di clustering delineando le caratteristiche di ogni cluster, tramite media, varianza e lista di valori degli attributi.
Prima di concludere, c'è da specificare che quando si parla di algoritmo o soluzione "migliore" lo si fa si sempre in relazione ad una indice di validità che offre solamente una stima della "qualità" del risultato. Questa stima non vuole in nessun modo sostituire l'importante collaborazione tra l'analista dei dati e l'esperto di mercato: entrambi, in maniera sinergica, dovranno valutare le soluzioni e prendere spunto da esse per la definizione finale dei segmenti di mercato.

1.3 Struttura della tesi

La tesi è strutturata nel modo seguente.
Nel Capitolo 2 analizzeremo la letteratura accademica sul clustering, con un particolare accento sul problema della dimensionalità. Inoltre, discuteremo le tecniche per la pulizia dei dati e le sfide sulla valutazione dei dataset e dei risultati del clustering.
Nel Capitolo 3 descriveremo il modulo RTP di Neosperience, per il quale

questo lavoro è stato svolto, e presenteremo la nostra soluzione, dalla raccolta dei dati fino alla esecuzione degli algoritmi
Nel Capitolo 4 discuteremo la procedura ed i risultati dell'analisi sperimentale.
Nel Capitolo 5 esporremo le conclusioni e i possibili sviluppi del lavoro.

Capitolo 2

Stato dell'arte

Il termine *clustering* si riferisce ad un insieme di tecniche che permettono la suddivisione di dati in gruppi (o *cluster*) di oggetti simili. In questo capitolo presentiamo alcune definizioni preliminari e analizziamo i passi principali della cluster analysis, dalla fase di preparazione dei dati fino all'identificazione di misure di validità per la valutazione dei risultati. Gli aspetti teorici descritti sono stati alla base del nostro lavoro di tesi e hanno portato alla realizzazione di numerosi strumenti di supporto alla segmentazione di mercato, cioè a quel processo che permette di partizionare un ampio mercato in piccoli gruppi di clienti caratterizzati da simili comportamenti d'acquisto.

2.1 Introduzione

L'*apprendimento automatico* (o *machine learning*) è una delle aree più importanti dell'intelligenza artificiale. Esso si occupa della realizzazione di algoritmi che, partendo da un insieme di dati, portano alla sintesi di nuova conoscenza. Possiamo individuare due grandi tipologie di apprendimento automatico: l'*apprendimento supervisionato* (o *supervised learning*) e l'*apprendimento non supervisionato* (o *unsupervised learning*) [67]. Il primo riguarda la progettazione di sistemi che generano una funzione capace di predire la *classe* di appartenenza di un dato in input sulla base di una serie di esempi già classificati. Al contrario, nell'apprendimento non supervisionato si progettano algoritmi che riorganizzano le informazioni in ingresso per identificare strutture naturalmente emergenti da esse. L'oggetto di studio di questo lavoro è la cluster analysis, che si colloca tra le tecniche di apprendimento non supervisionato.

2.1.1 Cos'è la cluster analysis

Viviamo in mondo pieno di dati. Ogni giorno le persone s'imbattono in una grande quantità di informazioni che vengono memorizzate per analisi successive. Uno dei modi principali per trattare questi dati è quello di classificarli in un insieme di categorie o cluster. La classificazione svolge da sempre un ruolo fondamentale per lo sviluppo scientifico dell'uomo che, per comprendere un nuovo fenomeno, cerca di confrontarlo con altri fenomeni noti basandosi sulla somiglianza tra questi. La cluster analysis, che ha come dominio di interesse i dati e non i fenomeni naturali, automatizza proprio questo processo, mettendo a disposizione una serie di strumenti per individuare informazioni nascoste o per dedurre modelli latenti all'interno di complesse strutture informative, che di per sé non mostrano nessuna regolarità. L'obiettivo principale da raggiungere è quello di creare dei raggruppamenti che massimizzino la similarità intra-cluster e, contemporaneamente, minimizzino quella inter-cluster, in modo tale che ogni gruppo esponga delle proprietà che lo qualificano in maniera determinante.
Le applicazioni della cluster analysis sono molto variegate. Ad esempio, in medicina è necessario identificare diversi tipi di tessuto nelle immagini PET [78], nelle scienze sociali è utile localizzare aree metropolitane con un'alta incidenza di criminalità [72], mentre nell'analisi delle reti sociali è interessante riconoscere comunità all'interno di un elevato numero di individui [66]. Di conseguenza, il concetto astratto di dato assume rispettivamente la forma di una immagine o di una sezione geografica o, ancora, di un nodo all'interno di un grafo.

2.1.2 Definizioni preliminari

Un dataset D indica un insieme di N dati (o *tuple*, *istanze*, *oggetti*) eterogenei. Ogni oggetto $x \in D$ è un vettore del tipo $x = (x_1, ..., x_m)$ dove l'i-esimo elemento x_i indica un attributo (o *dimensione*, *proprietà*) di x. Un attributo è *nominale* (o *categorico*) se ha un dominio finito i cui elementi rappresentano una categoria di appartenenza, oppure è detto *numerico* se ha un dominio, in generale, infinito i cui valori sono quantità numeriche [43].
Su due oggetti del dataset è possibile definire una *funzione di similarità* che misura quanto essi sono simili tra loro analizzando il valore degli attributi. La *matrice di prossimità* è una matrice simmetrica $N \times N$ in cui ogni elemento (i, j) rappresenta la similarità tra gli oggetti i e j.
Un *cluster* è un sottoinsieme $C \subseteq D$ i cui i suoi oggetti sono contemporaneamente simili tra loro e diversi dagli altri (secondo la definizione di similarità). Un *clustering* è un insieme di cluster. Più in particolare, nel *crisp clustering*

ogni oggetto appartiene ad uno e un solo cluster; nel *fuzzy clustering*, invece, un oggetto può appartenere a diverse comunità con un grado variabile mentre nell'*overlapping clustering* un oggetto appartiene a diverse comunità con lo stesso grado di appartenenza.
La *matrice di incidenza* è una matrice simmetrica $N \times N$ in cui ogni elemento (i, j) è uguale a 1 se i e j appartengono allo stesso cluster, oppure è pari a 0 se i e j appartengono a cluster diversi.
Un dataset può contenere oggetti, chiamati *outlier*, che non rispettano il comportamento generale o il modello sottostante dei dati. Gli outlier possono falsare sensibilmente il risultato degli algoritmi di clustering tanto da dover intervenire con una loro eliminazione tramite tecniche statistiche o tramite metodologie basate sul concetto di distanza [43].

2.1.3 Procedura di clustering

Per avere una idea d'insieme della cluster analysis analizziamone i quattro passi fondamentali.

Selezione o estrazione degli attributi La *feature selection* permette di selezionare un certo numero di proprietà rilevanti da un insieme di candidate, mentre la *feature extraction* utilizza delle trasformazioni per generare nuovi attributi da quelli originali [49]. L'assunto fondamentale alla base di queste tecniche è che i dati possono contenere molte proprietà ridondanti o non rilevanti: le dimensioni ridondanti sono quelle che non aggiungono nessuna informazione in più rispetto a quelle già selezionate, le feature non rilevanti non forniscono nessuna informazione in qualunque contesto. Gli effetti benefici di queste tecniche sono immediati: oltre a velocizzare gli algoritmi di mining, a ridurre la dimensione del dataset e a facilitare la presentazione dei risultati, migliorano significativamente la qualità e la granularità del clustering. Infatti, la suddivisione dei dati in gruppi omogenei è molto più semplice e "nitida" in uno spazio con poche dimensioni; inoltre, di frequente, i cluster emergono solamente in un sottoinsieme di attributi restando "nascosti" in uno spazio ad elevata dimensionalità (un gruppo di individui potrebbe essere simile per età, genere e luogo di nascita ma non per religione e città di residenza).
Questo è inoltre un passaggio fondamentale per evitare la cosiddetta *curse of dimensionality*, cioè una serie di complicazioni che si verificano con dati ad elevata dimensionalità. Ciò introduce due ordini di problemi: il primo riguarda la presenza di attributi non rilevanti ai fini del clustering che eliminano ogni tendenza al raggruppamento; il

secondo, invece, comporta il fatto che un qualsiasi punto del dataset risulta praticamente equidistante dal punto più vicino e da quello più lontano. Questo è un problema determinante perché, senza una oggettiva funzione di distanza, si perde il metro di giudizio per identificare due oggetti simili e individuare cluster significativi. Per un approfondimento su queste e altre problematiche si veda [56], [14] e [3].

Progettazione o selezione degli algoritmi di clustering Questo passo è usualmente accompagnato dalla scelta di una misura di similarità e di una funzione obiettivo: in particolare un algoritmo di clustering deve utilizzare la funzione di similarità per giudicare simili o dissimili due oggetti e deve disporre di una funzione obiettivo per decidere qual è il miglior raggruppamento. Ciò rende il problema delle ricerca dei cluster un problema d'ottimizzazione, ben definito matematicamente e con variegate soluzioni in letteratura.

Validazione del clustering Su uno stesso dataset, ogni algoritmo di clustering può sempre generare un raggruppamento, indipendentemente dal fatto che questo sia effettivamente significativo. Inoltre, algoritmi differenti portano a soluzioni differenti e, persino per lo stesso algoritmo, una diversa scelta dei valori dei suoi parametri può influire sul risultato finale. Per questa ragione è utile disporre di strumenti (detti *indici di validità*) che permettano di giudicare oggettivamente il grado di confidenza dei risultati del clustering. Essi si dividono in esterni ed interni. I primi si basano sulla presenza di un clustering specificato a-priori (*ground-truth*), che serve per valutare la qualità dell'algoritmo misurando quanto la sua soluzione si discosta dalla ground-truth. Gli indici interni invece non si basano su nessuna informazione esterna, ma esaminano la struttura del clustering direttamente dai dati originali. Per un compendio approfondito sugli indici di validità si faccia riferimento a [36], [50].

Interpretazione dei risultati L'ultimo obiettivo della cluster analysis è fornire agli utenti una rappresentazione immediata e facilmente comprensibile del risultato ottenuto, in modo tale da poter procedere con una valutazione soggettiva dell'utente: infatti, il ricorso ai soli indici non è mai sufficiente per misurare la qualità del clustering. L'affermazione *clustering is in the eyes of the beholder* [69] conferma la tesi secondo cui la decisione definitiva sulla coerenza del risultato deve essere presa dall'osservatore. Verranno quindi consultati esperti del do-

minio di interesse per interpretare, convalidare e, semmai, trasformare le soluzioni proposte dagli algoritmi.

2.1.4 Segmentazione di mercato

La *segmentazione di mercato* è definita come quel processo che permette di partizionare un ampio mercato in piccoli gruppi di clienti (o *segmenti*) caratterizzati da bisogni omogenei e simili comportamenti d'acquisto [71], [85]. Conoscere questi segmenti è utile ai dirigenti aziendali non solo per creare soluzioni ad-hoc per ogni tipologia di cliente ma anche per definire con elevata precisione strategie competitive di mercato. I segmenti sono costruiti sulla base di caratteristiche demografiche e geografiche, considerando anche le vendite avute in passato e le informazioni provenienti da sondaggi sul prodotto [9], [1].
Con l'avvento dei Big Data si assiste a due fenomeni fondamentali: da un lato è molto più semplice individuare i propri clienti grazie all'utilizzo dei Social Network, dall'altro, ogni cliente è descritto da un elevato numero di attributi, come quelli recuperabili dal suo profilo Facebook. Questa esplosione di informazioni rende più appetibile e potenzialmente più proficuo il processo di segmentazione, ma aggiunge, d'altro canto, un ulteriore livello di difficoltà. Gestire un grande insieme di clienti e individuare informazioni nascoste al suo interno diventa davvero difficoltoso senza l'ausilio di un procedimento automatico. È proprio a questo punto che la cluster analysis interviene a favore della segmentazione di mercato, fornendo una serie di strumenti che agevolano l'identificazione dei caratteri emergenti dalla base di dati aziendale.
Vediamo ora, nello specifico, quali sono state le tecniche studiate e utilizzate per il raggiungimento degli obiettivi prefissati.

2.2 Data Preprocessing

Le basi di dati odierne, a causa della loro elevata dimensione, contengono spesso informazioni altamente rumorose, mancanti o incoerenti. Inevitabilmente, la bassa qualità dei dati porta ad avere una bassa qualità dei risultati di mining. Con il termine *data preprocessing* s'intende una serie di tecniche e strumenti che aiutano a migliorare la qualità dei dati: il *data cleaning* (per rimuovere rumore o correggere inconsistenze), il *data integration* (per combinare dati provenienti da fonti diverse e creare un'unica sorgente coerente), il *data reduction* (per ridurre la dimensione della base di dati aggregando o eliminando caratteristiche ridondanti) e il *data transformation* (che consen-

te di trasformare i dati da un formato di partenza ad uno di destinazione). Tutte queste tecniche vengono usate in maniera congiunta per ottenere una base di dati coerente che possa essere usata con efficacia nel processo di clustering.
In particolare, tra queste tecniche spiccano tre fondamentali metodologie: la selezione di attributi rilevanti, il rilevamento e la rimozione di outlier e l'inferenza dei valori mancanti.

Selezione degli attributi

Come già affermato nei paragrafi precedenti, la feature selection permette di selezionare un certo numero di proprietà rilevanti da un insieme di candidate.[1] Gli algoritmi di feature selection possono essere classificati secondo la tipologia di output che generano: da un lato vediamo quelli che producono una lista di attributi ordinati in base ad un peso di rilevanza; dall'altro quelli che escludono gli attributi considerati insignificanti. In particolare, ogni algoritmo può essere supervisionato o non supervisionato a seconda del fatto che sia disponibile o meno una informazioni esterna: l'*etichetta di classe*. Basandosi su questa informazione aggiuntiva, è possibile determinare quali sono le feature essenziali che spiegano al meglio i valori dell'etichetta di classe: se ci si accorge che i valori di una certa dimensione sono scorrelati da quelli dell'etichetta, allora quella dimensione verrà eliminata [42], [70].
Tecniche non supervisionate che hanno riscontrato un buon successo teorico e pratico si basano sul clustering degli attributi: il loro output è costituito dalla selezione delle dimensioni più rappresentative di ogni cluster [68]. Altri approcci interessanti come [98] si basano invece sulla definizione preliminare della matrice di similarità S del dataset. Una feature è detta *coerente* se assegna valori simili alle istanze che sono vicine le une alle altre in S. Per un compendio completo su tecniche di feature selection (sia supervisionate che non) si faccia riferimento a [70], [23].

Outlier

Le tecniche di outlier detection vengono utilizzate per identificare e rimuovere osservazioni anomale dai dati. Le più semplici sono basate sul concetto di distanza e tentano di individuare gli outlier definendoli come quei punti sostanzialmente più "lontani" dagli altri. Ad esempio, nel DB(ϵ, π), [54] un

[1] In questo lavoro non abbiamo ritenuto vantaggioso utilizzare tecniche di feature extraction in quanto queste comportano una trasformazione delle dimensioni originali e di conseguenza portano inevitabilmente alla perdita di informazioni necessarie alla caratterizzazione della semantica dei risultati.

punto p è considerato un outlier se, al massimo una percentuale π di punti ha una distanza minore di ϵ da p.
Altri algoritmi lavorano sulla densità [20] [19] (un outlier ha una densità "locale" molto più bassa della densità della maggior parte degli altri punti); altri ancora ricostruiscono preliminarmente un clustering sui dati e definiscono outlier quei punti che appartengono a gruppi di piccole dimensioni [46], [22].

Valori mancanti

In una collezione di dati è molto probabile che il valore di certi attributi sia mancante; ciò comporta inevitabilmente la necessità di dover adottare una metodologia che gestisca l'assenza di questi valori. Le tecniche di inferenza statistica più semplici prevedono di sostituire il valore mancante di una variabile con la media totale degli altri valori. Spesso queste tecniche producono un errore di stima aggiungendo un *bias* alla base di dati. Si pensi, infatti, all'inferenza dell'età di un individuo tramite l'età media dei suoi amici: se questo attributo possiede un'alta varianza o è mancante in un gran numero di individui allora il valor medio non è significativo [82]. Per ovviare a questi e altri inconvenienti si utilizzano delle tecniche più complesse che si basano sulla tipologia dei dati mancanti [27]. Quando la probabilità che una osservazione sia mancante è indipendente da ogni altra caratteristica dell'individuo si parla di dati *Missing Completely At Random* (MCAR). In questo caso si riesce a produrre una inferenza oggettiva anche tramite tecniche basilari. Se, invece, la probabilità che una osservazione sia mancante dipende da informazioni non osservate, come il valore dell'osservazione stessa, i dati vengono detti *Missing Not At Random* (MNAR). In questo caso, quindi, la mancanza di una osservazione non è completamente casuale ma dipende da una caratteristica non osservata dell'individuo: ad esempio, in un questionario, un uomo ricco potrebbe non rispondere a domande riguardanti il suo salario. Qui, poiché l'informazione mancante (il salario) dipende da qualche variabile non osservata (la ricchezza), è molto difficile non introdurre errori di stima con qualsiasi tecnica di inferenza. Infine, quando la probabilità che un valore non sia specificato dipende da altre variabili osservate (ma non dal valore stesso) si parla di dati *Missing At Random* (MAR). In questo caso le tecniche triviali di inferenza spesso producono risultati scorretti [37], [90], quindi è necessario utilizzare metodologie più sofisticate. Un esempio è dato dall'algoritmo del *donatore di minima distanza*, in cui si sceglie di sostituire il dato mancante in un individuo con quello presente nell'individuo ad esso più simile. Altre metodologie si avvalgono di potenti strumenti statistici

(regressione, algoritmi di *expectation-maximization*, inferenza multipla). Per una analisi approfondita si rimanda a [61], [83], [82] e a [55] per l'inferenza di dati in reti sociali.

2.3 Classificazione degli algoritmi di clustering

La categorizzazione degli algoritmi di clustering non è univoca, e gli algoritmi possono travalicare i confini tra approcci diversi; in questa sezione presentiamo la divisione canonica delle soluzioni al clustering, in base alla natura dei cluster generati e alle tecniche per ottenerli.

2.3.1 Metodi gerarchici

Il clustering gerarchico costruisce una gerarchia di cluster, un albero binario detto *dendrogram* avente alla radice l'intera popolazione e alle foglie i singoli individui. Un nodo intermedio costituisce pertanto un cluster dei propri discendenti, e tagliando l'albero a varie altezze si ottengono soluzioni di diversa dimensione al problema del clustering, con un numero crescente di gruppi man mano che ci si allontana dalla radice. Il clustering gerarchico offre una flessibilità nativa nel livello di granularità della soluzione; soprattutto, è compatibile con qualsiasi definizione di similarità e conseguentemente applicabile ad ogni tipo di dato. D'altronde, le principali critiche a questa tecnica risiedono nella mancanza di robustezza che rende la soluzione vulnerabile a rumore e anomalie, nell'inabilità di rivedere le scelte effettuate e correggere eventuali errori di classificazione nei cluster intermedi ed infine nella tendenza a formare cluster sferici. Tuttavia, il limite principale del clustering gerarchico, che preclude la sua adozione per problemi di clustering in grande scala, è la complessità almeno quadratica. Le implementazioni di maggiore rilievo sono CURE (dati numerici) [40], ROCK (dati categorici) [39], BIRCH (ottimizzato per dataset di ampie dimensioni) [97] e Chameleon [52].

2.3.2 Metodi partitivi

Gli algoritmi partitivi dividono i dati in sottoinsiemi privi di una struttura gerarchica. Poiché identificare la soluzione ottima dall'enumerazione di tutte le possibili partizioni è notoriamente NP-difficile, sono stati sviluppati algoritmi euristici: partendo da una funzione obiettivo, la soluzione è costruita iterando un problema di ottimizzazione, riassegnando gli individui tra i cluster al fine di massimizzare il punteggio del clustering finale. La differenza sostanziale tra algoritmi partitivi è gerarchici è proprio questo

meccanismo di rilocazione, che corregge eventuali errori di 'classificazione' degli individui.
Gli algoritmi partitivi possono essere ulteriormente caratterizzati:

centroid-based: specificato il numero K di cluster, gli algoritmi basati sulla distanza producono un vettore di K centroidi—punti arbitrari nello spazio N-dimensionale dei dati—ed un assegnamento tra individui e centroidi, tale che la somma dei quadrati delle distanze tra individui e centroidi sia minima. Questa tipologia di algoritmi risente fortemente della scelta iniziale dei centroidi; inoltre produce cluster di forma sferica. Algoritmi notevoli sono *K-means*, *K-medoids*, *Fuzzy c-means*; limitatamente agli algoritmi che abbiamo considerato per questo lavoro, si veda [26].

model-based: nella visione probabilistica, ogni cluster presente nei dati è espressione di una distribuzione parametrica di probabilità; detto altrimenti, i dati sono concettualmente ottenuti campionando una mescolanza di distribuzioni o *mixture model*, e lo scopo del clustering è stimare dai dati i parametri di tali distribuzioni latenti. Il criterio di ottimalità è pertanto la verosimiglianza (*likelihood*), che esprime la capacità dei parametri del modello di spiegare i dati. L'algoritmo canonico di questa tipologia è *Expectation-Maximization* (EM); limitatamente agli algoritmi che abbiamo considerato per questo lavoro si vedano [11], [96] e [44].

density-based l'idea alla base di queste tecniche è che un cluster sia una regione sensibilmente più densa di punti rispetto allo spazio circostante. Conseguentemente, i cluster sono separati da regioni rarefatte dello spazio, ed i punti che ricadono in queste aree sono solitamente rumore o anomalie, che vengono quindi identificati e scartati automaticamente. Poiché i cluster si espandono nella direzione lungo la quale la densità cresce, gli algoritmi basati sulla densità scoprono cluster di forma arbitraria; in ultimo, scalano bene con il volume dei dati. Tuttavia questo approccio presenta anche un evidente limite: la densità nello spazio è funzione dalla distanza tra i punti, che, come già affermato, perde di significatività al crescere delle dimensioni. Il più noto algoritmo in questa categoria è DBSCAN [29].

grid-based i metodi discussi precedentemente sono guidati dai dati, in quanto partizionano l'insieme degli oggetti in accordo con la distribuzione dei dati nello spazio. Alternativamente, gli algoritmi grid-based suddividono lo spazio in una griglia multidimensionale a risoluzione

variabile; poiché le successive operazioni di clustering saranno eseguite sulle celle di tale griglia, le prestazioni sono indipendenti dalla dimensione dei dati ma dipendenti unicamente dal numero di celle in ciascuna dimensione dello spazio quantizzato. I rappresentanti più noti di questa categoria sono CLIQUE [5] e la sua evoluzione, MAFIA [35], e OPTIGRID [45].

2.3.3 Subspace Clustering

Il *subspace clustering* [77],[69] è una estensione del clustering tradizionale che cerca di determinare la presenza di possibili cluster in tutti i sottospazi[2] dell'insieme di attributi. Spesso, quando si ha a che fare con dati ad alta dimensionalità, molti attributi risultano essere irrilevanti e possono nascondere l'effettiva presenza di un naturale raggruppamento [86]. Se da un lato la feature selection sceglie per ogni cluster lo stesso insieme di attributi rilevanti, il subspace clustering associa ad ogni cluster un diverso sottospazio, ossia una diversa selezione di attributi significativi [77].
Ad esempio l'algoritmo $CLIQUE$ [5], uno dei primi algoritmi progettati per la ricerca di cluster nei sottospazi, si basa su una interessante osservazione: una regione che è densa in un particolare sottospazio sarà necessariamente densa una volta proiettata in un sottospazio di dimensione inferiore. Il contrario, invece, non è necessariamente vero. Basandosi su questa intuizione, l'algoritmo parte analizzando sottospazi densi con poche dimensioni che possono solo suggerire la presenza di cluster con dimensioni più elevate; detto altrimenti, se una regione a m dimensioni non è densa allora non lo è neppure una con $m^{'} > m$ dimensioni. Questo ragionamento può essere utilizzato per identificare unità dense in un qualunque sottospazio, impiegando, quindi, una procedura molto più efficiente dell'ispezione completa di tutti i possibili sottospazi.

2.3.4 Ensemble Clustering

E' noto che un singolo problema può essere risolto con tecniche diverse; ogni tecnica, d'altro canto, proporrà una soluzione con una qualità variabile a seconda delle caratteristiche con cui è stata progettata. I metodi *ensemble*, invece di adottare un'unica metodologia per la risoluzione di un problema, utilizzano un insieme di strumenti con l'obiettivo di disporre di un ventaglio di soluzioni diverse. Il vantaggio di questo approccio consiste, ad esempio,

[2] Per una definizione precisa di sottospazio si rimanda a [91]. Qui ci basta intuire che un sottospazio è un sottoinsieme degli attributi originali.

nel poter scegliere il risultato più comune (*majority voting*) o nel combinare i differenti output in un'unica soluzione finale. Tipicamente i metodi ensemble sono impiegati nell'ambito dell'apprendimento supervisionato in cui si creano diversi *training set* per la costruzione di altrettanti classificatori. Se da un lato il vantaggio principale risiede nel fatto di ottenere sensibili miglioramenti nelle performance predittive, dall'altro lo svantaggio più evidente è quello di dover interpretare e unificare le singole soluzioni [25], [80]. Nel dominio dell'apprendimento non supervisionato, e in particolare nel clustering, sono stati sviluppati dei procedimenti che ricalcano i concetti appena esposti: invece di utilizzare un solo algoritmo di clustering che produrrà, tipicamente, un unico raggruppamento dei dati, vengono adottati un insieme di algoritmi che porteranno, quindi, a partizionamenti multipli. Il problema di fondo, come di norma, è quello di interpretare e combinare le soluzioni. Un approccio di base consiste nell'assegnare un punteggio ad ogni coppia di punti i e j del dataset. La coppia riceverà un punteggio massimo o minimo a seconda del fatto che tutti gli algoritmi di clustering li abbiano considerati o meno appartenenti allo stesso raggruppamento. In questo modo si dispone, per ogni coppia di punti, di un indice di similarità che può essere utilizzato come input di un ulteriore algoritmo basato sulla distanza. Si faccia riferimento a [87], [33] per un approfondimento sulle tecniche di ensemble.

2.3.5 Clustering di grafi e reti sociali

Definizioni

Un *grafo* è una struttura matematica usata per modellare relazioni tra oggetti. Esso può, ad esempio, rappresentare l'amicizia all'interno di un gruppo persone, oppure la struttura bio-molecolare di DNA e proteine [89], [64]. Quando si associa ad ogni oggetto (o *vertice*) un insieme di proprietà che lo descrivono si parla di grafo con attributi [99], [95]. Particolarmente interessante è l'approccio di [99] nel gestire un grafo con attributi: se nel dataset è presente una coppia attributo-valore (a, v) allora viene aggiunto, nel grafo originale, un nodo $n_{a,v}$ che sarà collegato a tutti i vertici che presentano il valore v per l'attributo a. In questo modo si crea un grafo *aumentato* su cui è possibile definire funzioni obiettivo e di similarità che tengano conto, contemporaneamente, della struttura topologica del grafo e dei suoi attributi.
Una *rete sociale* è un grafo costituito da *individui* (o *attori*) connessi da una relazione sociale. Queste reti hanno delle proprietà ampiamente studiate in sociologia che rivestono un ruolo fondamentale anche nella cluster analysis[3]:

[3] In questo contesto è più preciso parlare di *comunità* invece che di cluster e di algoritmi di *community detection* invece che algoritmi di clustering.

omofilia, *transitività* e *clustering locale*. L'omofilia rappresenta la tendenza secondo cui è molto probabile che ci sia una relazione sociale tra individui simili [65], [59]. La transitività afferma che se due attori condividono un legame sociale con un terzo attore allora è verosimile che essi stessi siano tra loro relazionati [94]. Il clustering locale, infine, indica la presenza di un'alta densità di relazioni localmente ad alcuni individui della rete: ciò spiega, ad esempio, il fatto che i nostri amici sono anche amici tra di loro [93], [73].

Clustering su grafi

Per definire una comunità dal punto di vista topologico, ci si può basare sull'intuizione che al suo interno devono esserci molti più archi di quelli che ci sono tra essa e il resto del grafo. Ad esempio, le comunità sociali possono essere definite come dei sottogruppi i cui membri sono tutti amici tra loro [63], e questo, in termini di grafi, corrisponde ad una *clique*, cioè ad un sottoinsieme di vertici tutti collegati gli uni con gli altri. Parallelamente, una comunità può essere definita utilizzando una funzione di *fitness* che misura quanto essa sia coesa internamente. Una delle funzioni più semplici è la *densità intra-cluster*, che esprime il rapporto tra il numero di archi interni ad un sottografo e il numero di tutti gli archi che questo potrebbe avere. Per esempio, se un certo sottografo ha una densità maggiore di una soglia ϵ allora può essere considerato una comunità[4]. Il passo successivo, come al solito, consiste nella progettazione di una funzione di similarità tra nodi e di una funzione che indichi la qualità della divisione della rete. Una misura vastamente utilizzata in letteratura è la *modularità*. Essa si basa sul concetto di *modello nullo* di un grafo G che, secondo Girvan e Newman [74], rappresenta una copia di G in cui gli archi sono riconfigurati in maniera completamente casuale, con il vincolo che il grado atteso di un nodo sia pari al grado che il nodo stesso possiede in G. Per questa ragione il modello nullo presenta la fondamentale caratteristica di non possedere una struttura a comunità [30]. In definitiva, la modularità indica quanto si discosta il grado di coesione di un certo partizionamento dal grado di coesione dello stesso partizionamento nel modello nullo.
La maggior parte degli algoritmi di clustering su grafi, in estrema sintesi, definiscono prima la struttura di una comunità e, poi, in maniera algoritmica

[4] Trovare una clique o sottografi che hanno una densità intra-cluster maggiore di una certa soglia sono entrambi problemi **NP**-difficili [16], [32]. Dal punto di vista pratico si rilassano certe condizioni in modo tale da avere a disposizione soluzioni computazionalmente più efficienti [7], [10]

e incrementale, tentano di identificare i cluster nel grafo, massimizzando la modularità [15].

Campionamento di grafi

In molte applicazioni si ha la necessità di eseguire complicati algoritmi su grafi: dalla simulazione dei protocolli di routing all'analisi degli scenari del *viral markenting*. Spesso, però, si ha a che fare con grafi di dimensioni elevate che renderebbero l'esecuzione di tali algoritmi non tollerabile dal punto di vista computazionale. Per questa ragione sono state sviluppate numerose tecniche di campionamento in cui il grafo campionato riflette le caratteristiche topologiche di quello originale. Le strategie più semplici sono quelle che si basano sul campionamento casuale di nodi (*random node selection*) o di archi (*random edge selection*). Tecniche più complicate utilizzano, invece, il concetto di *esplorazione*: l'idea di fondo è quella di selezionare un nodo iniziale in maniera casuale e poi esplorare i nodi che sono ad esso vicini. Ad esempio, nel campionamento basato sui *random walk* si sceglie casualmente un nodo iniziale e da questo si simula una random walk sul grafo: ad ogni passo o si seleziona un vicino del nodo precedente oppure, con una probabilità c, si ricomincia dal nodo iniziale. Un'altra tecnica è il *forest fire sampling* in cui, si seleziona un vertice iniziale (o *seed*) e si iniziano a "bruciare" gli archi uscenti dal seed. Se un arco è stato bruciato allora il vertice che si trova all'altra estremità ha la possibilità, a sua volta, di bruciare i propri archi, e così via di seguito [60]. Altre tecniche interessati sono studiate in [57], [2], [6].

2.4 Indici e misure

2.4.1 Metriche topologiche

Al fine di studiare l'effetto delle connessioni tra gli individui sulla qualità del clustering, è necessario catturare numericamente le sfaccettature della topologia. In prima approssimazione, un grafo è una collezione di nodi e archi, ed esiste una plausibile dipendenza tra il volume dei dati—il numero di nodi e archi—e la bontà del risultato, in funzione dell'algoritmo scelto. Le più semplici misure di connettività del grafo sono gli indici beta e gamma, rispettivamente il rapporto tra nodi e archi ed il rapporto tra il numero di archi esistenti e il massimo numero teorico di archi in un grafo delle stesse dimensioni. Similmente, la transitività o coefficiente di clustering globale misura la probabilità che nodi con un vicino in comune siano tra loro connessi [92].

Avendo studiato le proprietà teoriche delle reti sociali, è interessante quantificarle in un grafo sotto esame. L'assortatività misura il grado di omofilia in una rete, cioé la preferenza di un nodo a connettersi a nodi che gli assomigliano. Limitatamente alla topologia, l'assortatività può essere calcolata usando come criterio di similarità il grado di un nodo, ovvero il numero di archi incidenti. In genere, le reti sociali esibiscono alti valori di assortatività [75, 76]. Un'altra caratterizzazione delle reti sociali è ottenuta dalla distribuzione del grado dei vertici. Una rete è detta *scale-free* se la distribuzione di probabilità dei gradi segue una legge di potenza—ovvero la frazione di nodi aventi grado k è proporzionale a $k^{-\gamma}$—per un esponente $2 < \gamma < 3$. Nelle reti sociali, questa proprietà prende il nome di *preferential attachment*[12]. Stimando l'esponente γ dai dati è possibile ricavare un altro indice della 'socialità' della rete.
La centralizzazione è un metodo per sintetizzare a livello di grafo i punteggi di centralità—l'importanza relativa all'interno del grafo—dei singoli vertici [31, 92]. Abbiamo considerato diverse misure di centralità a livello di nodo: il grado, i cammini minimi, l'intermedietà, gli autovettori della matrice di adiacenza. La prossimità o *closeness* è funzione dal numero di passi necessari a raggiungere un qualsiasi altro nodo partendo dal vertice dato, e decresce man mano che ci si avvicina alla periferia della rete. L'intermedietà o *betweenness* quantifica il numero di cammini minimi che attraversano il nodo, e descrive quanto un nodo sia un accentratore o *hub* di relazioni sociali. La centralità degli autovettori o *eigenvector centrality* [17] assume che la centralità di un nodo sia proporzionale alle centralità dei nodi a cui è connesso; in generale, i nodi con un punteggio elevato sono quelli connessi a molti altri nodi che, a loro volta, sono connessi a molti altri (e così via).
Complementare alla connettività è la frammentazione, cioè quanto un grafo si discosta dall'essere omogeneamente coeso e completamente connesso. La misura più ovvia di frammentazione di un grafo è il numero di componenti connesse o sottografi massimali; tuttavia questa misura trascura tanto la dimensione delle componenti—singoli vertici isolati dovrebbero contribuire poco alla frammentazione—quanto la struttura interna delle componenti. In considerazione della dimensione delle componenti abbiamo adottato la misura F [18] e l'entropia; per valutare il grado di coesione nelle componenti, si è fatto ricorso alla *distance fragmentation* [18]. Queste misure sono tutte statiche, ossia realizzano un istantanea della frammentazione del grafo trascurando la stabilità delle componenti connesse esistenti. Per valutare la robustezza di un grafo abbiamo studiato l'evoluzione della misura F man mano che i nodi più importanti della rete sono rimossi; in questo scenario, l'importanza di un nodo è la sua intermedietà o alternativamente la *bridging*

centrality [47].

2.4.2 Indici di validità del clustering

Come già affermato, una delle sfide fondamentali della cluster analysis è valutare i risultati degli algoritmi senza disporre di informazioni ausiliarie. Un approccio comune vede l'utilizzo dei già citati indici di validità che si dividono in indici esterni (che valutano il raggruppamento basandosi su una struttura pre-specificata, ground truth) e indici interni (che si basano solamente sull'informazione intrinseca dei dati). La loro scelta deve essere preceduta da un'attenta analisi dell'algoritmo di clustering, in quanto quest'ultimo potrebbe ottimizzare proprio gli stessi parametri misurati dall'indice e ciò comporterebbe una valutazione non oggettiva dei risultati. Gli indici esterni non sono oggetto di studio del presente lavoro e quindi nominiamo solamente la *F-measure*, la *purezza* e l'*entropia* che è possibile approfondire qui [79].
Riguardo gli indici interni citiamo e descriviamo quelli che sono stati utilizzati in questa tesi. La tecnica del *Silhouette index* [81], ad esempio, assegna ad ogni oggetto x del dataset un punteggio che, tramite una funzione di similarità, calcola quanto esso sia appropriatamente raggruppato: se x è simile agli oggetti del cluster cui appartiene e dissimile da tutti gli altri allora riceverà un punteggio elevato. La media dei punteggi rappresenta l'indice Silhouette. Un'altra tecnica che è stata ampiamente utilizzata si basa sul calcolo della correlazione tra la matrice di prossimità e la matrice di incidenza (si veda la sottosezione 2.1.2 per le definizioni). La correlazione varia nell'intervallo $[-1, 1]$, in cui valori prossimi ad 1 (-1) indicano che i punti appartenenti allo stesso cluster sono molto simili (dissimili) tra loro; valori vicini allo 0 indicano, sostanzialmente, la presenza di un clustering casuale.
Per concludere ricordiamo la già citata *modularità* che può essere considerata, a tutti gli effetti, un indice di validità interno.
C'è da precisare che questi indici non hanno un valore dogmatico ma offrono solo un suggerimento utile all'osservatore che potrà avvalersi di uno strumento in più per la valutazione soggettiva dei risultati. Detto altrimenti, la segmentazione di mercato non deve ritenersi risolta con lo studio degli indici di validità ma deve continuare con una valutazione personale del decision maker, che stabilirà l'effettiva legittimità dei risultati o ne modificherà la struttura basandosi anche su variabili esterne (come profili socio-demografici e geografici), che non hanno mai preso parte al processo di clustering.

Misure di similarità

Come già affermato nei paragrafi precedenti, un passaggio fondamentale della cluster analysis è la definizione di una misura di similarità tra gli oggetti del dataset. La scelta di questa funzione deve essere fatta accuratamente tenendo conto di vari fattori: (1) della *tipologia di attributi* (una similarità pensata per dati numerici, come quella euclidea, non avrebbe senso su dati categorici); (2) della *dimensionalità* dei dati, in quanto, come affermato in [3], in presenza di un alto numero di dimensioni, alcune definizioni di similarità potrebbero essere non significative; (3) della presenza di missing value, poiché ci si potrebbe trovare a dover confrontare due istanze con un numero diverso di attributi: in questo caso, ad esempio, è possibile tener conto solo degli attributi comuni oppure trattare i due vettori di attributi come degli insiemi. Limitatamente a quanto impiegato nel presente lavoro, definiamo e descriviamo alcune misure interessanti.

Indice Jaccard L'indice Jaccard [48] misura la similarità tra due insiemi di oggetti ed è definito come il rapporto tra la cardinalità della loro intersezione e la cardinalità della loro unione. Viene utilizzato su dati categorici e risulta essere intrinsecamente flessibile in caso di valori mancanti. Il suo intervallo è $[0, 1]$.

Coseno di similitudine Dati due vettori numerici di uguale dimensione, il coseno di similitudine è un numero compreso tra -1 e 1 ed è definito come prodotto scalare dei vettori normalizzati. Risulta essere molto efficace su dati con molte dimensioni [28]

Distanza di Mahalanobis La distanza di Mahalanobis misura la dissimilarità tra due vettori numerici di uguale dimensione. È una generalizzazione della distanza euclidea in quanto tiene conto della matrice di covarianza del dataset: infatti, se la matrice di covarianza è pari alla matrice identità si riduce alla distanza euclidea. Viene spesso utilizzata per l'individuazione degli outlier all'interno di dataset dalla forma ellittica mostrando performance superiori alla distanza euclidea [34].

Distanza Frazionaria La distanza frazionaria risulta essere molto efficace su dati con molte dimensioni, migliorando significativamente la qualità degli algoritmi di clustering distance-based [3]. Intuitivamente è molto simile alla distanza euclidea con la particolarità di elevare al quadrato invece di usare la radice quadrata e, viceversa, usare la radice quadrata

al posto di elevare al quadrato[5]. È applicabile su dati numerici con ugual numero di dimensioni.

[5]In questo caso è, forse, molto più immediata leggere la formula analitica di questa misura per comprendere anche la completa generalizzazione: se m è il numero di dimensioni del dataset, la distanza frazionaria tra due vettori x e y è calcolata come: $[\sum_{i=1}^{m}(x_i - y_i)^f]^{1/f}$ con $f \in (0, 1)$.

Capitolo 3

Architettura del framework

In questo capitolo descriviamo l'architettura del framework sviluppato durante il lavoro di tesi. Inizialmente presentiamo la piattaforma *Neosperience* specificando dove si colloca il nostro lavoro al suo interno e poi mostriamo quali sono i moduli progettati e le soluzioni proposte per ogni fase della data analysis. Vengono affrontate quindi le sfide di raccolta, preparazione e conversione dei dati, fino all'esecuzione degli algoritmi di clustering. Infine, presentiamo un *workflow* modulare dell'intera architettura, utile per una visione d'insieme degli strumenti disponibili.

3.1 Neosperience RTP

Neosperience è una azienda che sviluppa strumenti software per la *Digital Customer Experience*. In un mercato globale iper-competitivo e iper-connesso, per contrastare la crescente dispersione degli utenti, un marchio deve stabilire un legame individuale con il cliente. Customer experience descrive la percezione che il cliente matura—razionalmente ed emotivamente—della relazione con il marchio. Di conseguenza, la gestione dell'*esperienza cliente* è la pianificazione e la risposta alle interazioni con il consumatore, al fine di raggiungerne ed eccederne le aspettative ed in questo modo aumentare la soddisfazione, la lealtà e il sostegno del cliente al marchio[1]: in una parola, coinvolgimento o *engagement*. L'esperienza cliente *digitale* apre nuove vie di contatto con le persone: non solo portali internet, ma applicazioni per social network e dispositivi mobili, tecnologie 3D e negozi virtuali, realtà aumentata e *gamification*. Il ciclo vitale di un cliente rispetto ad un mar-

[1] "The practice of designing and reacting to customer interactions to meet or exceed customer expectations and, thus, increase customer satisfaction, loyalty and advocacy" - Gartner

chio inizia con la scoperta di un prodotto o un servizio; dopo questo primo contatto, seguono diversi *moment of truth*, le occasioni in cui il cliente interagisce con l'azienda e si forma una opinione, consapevole ed inconscia. Il mantenimento del cliente dipende dalla capacità del marchio di influire positivamente in ciascuno di questi istanti: la valutazione del prodotto, la decisione dell'acquisto e l'esperienza d'uso. Questo non è solo qualità del servizio, ma arrivare ad una comprensione così profonda dell'utente da poter offrire una esperienza—contenuti e benefici—tanto personalizzata ed appagante da indurlo non solo a restare leale al marchio ma a convincere altri ad avvicinarsi. Per raggiungere questo grado di conoscenza è necessario estrarre indizi da ogni punto di contatto con il cliente, sfruttando l'immensa mole di informazione nella Rete.
Recentemente, Neosperience ha aggiunto alla propria piattaforma di customer experience il modulo Right-Time Personalization (RTP), per offrire a ciascun cliente contenuti personalizzati, scelti e presentati in base ai suoi interessi e intenzioni di acquisto. Questo modulo potrebbe essere integrato in un negozio virtuale o una galleria di contenuti, laddove ad ogni prodotto può essere associato un target ovvero il destinatario più appropriato, definito mediante variabili demografiche (età, genere, luogo di nascita, istruzione), interessi e contesto (ad esempio la distanza dal negozio fisico). La piattaforma Neosperience, sottostante all'applicazione, recupera i dati personali dell'utente (da Facebook, Twitter e Foursquare) delineando quindi un profilo unico che lo contraddistingue. Partendo da questo profilo, RTP misura, per ogni oggetto nel catalogo, la somiglianza tra il target dell'oggetto e l'utente dell'applicazione, in modo tale che i prodotti vengano riordinati e visualizzati in ordine di significatività.
Il nostro lavoro è stato motivato dall'esigenza di Neosperience di valorizzare i dati degli utenti raccolti tramite il proprio modulo RTP. Questi dati infatti racchiudono il potenziale per un diverso approccio alla segmentazione della clientela, facendo leva su informazioni variegate, semi-strutturate e non strutturate, ma nuove e complementari rispetto alle tradizionali categorie sociali e geografiche. L'opportunità offerta da questi dati è la conoscenza puntuale degli interessi, delle opinioni, del *mood* di ciascun cliente, e la possibilità di rispondervi in tempo reale.

3.1.1 Il modello dei dati

Nel progetto originale, il modulo RTP è alimentato da tre sorgenti di dati: Facebook, Twitter e Foursquare.
Facebook, con oltre un miliardo di utenti mensili, è il più diffuso e noto

social network al mondo. In Facebook, un utente può pubblicare la propria storia personale, interessi, esperienze, foto, lavoro e perfino stati d'animo. D'altronde, il fulcro dei social network è tessere relazioni, e Facebook non fa eccezione: ogni utente può stringere amicizia con altri utenti, condividere con essi contenuti, conversare, giocare, organizzare eventi ed essere informato di ogni cambiamento nella propria rete sociale. Fra le tre sorgenti, Facebook contribuisce la porzione dominante di informazione: il profilo utente, i like, le amicizie. Come riportato in Figura 3.1, la frazione osservabile del profilo include il nome, il genere, la data e il luogo di nascita, la città di residenza, l'istruzione (scuola superiore, università e formazione specialistica), lo stato sentimentale e l'orientamento sessuale. Al pari di ogni altro nodo del grafo sociale di Facebook, i like sono identificati univocamente e organizzati in oltre 200 macrocategorie, talvolta ulteriormente frazionate in sottogruppi: questo ha permesso di ridurre notevolmente la dimensionalità del problema.

Twitter è un social network e una piattaforma di microblogging. Al cuore di Twitter vi sono i *tweet*, brevi messaggi in 140 caratteri che un utente pubblica e la piattaforma notifica a tutti i suoi contatti, chiamati *follower*. Un utente riceve i tweet delle persone che segue sulla propria bacheca o *home timeline*, da cui può rispondere, inserendosi nel flusso di tweet esistente, o ripubblicare (*retweet*), propagando il tweet ai propri follower. Per organizzare tematicamente le conversazioni, i tweet sono spesso etichettati con un *hashtag*, una parola chiave preceduta da un cancelletto, o *hash* in inglese. Questa convenzione nacque spontaneamente tra gli utenti di Twitter e fu raccolta ed integrata nella piattaforma, che oggi pubblica in tempo reale la lista dei *trending topics*, le parole o argomenti di discussione che compaiono più frequentemente negli ultimi tweet.

Foursquare è tanto un social network focalizzato sulla posizione dell'utente quanto un gioco, il cui successo è indubitabilmente connesso al dilagare di dispositivi mobili e intelligenti, sempre più comunemente provvisti di una antenna GPS. Foursquare non solo rende pubblica la posizione dell'utente, permettendo di trovare gli altri utenti nella medesima zona, ma invita gli iscritti a registrarsi (*check in*) ai luoghi di incontro che visitano—un negozio, un bar, un museo—e attribuisce punti e distintivi (*badges*) per premiare la frequenza con cui vi si accede o la scoperta di nuovi luoghi prima dei propri amici.

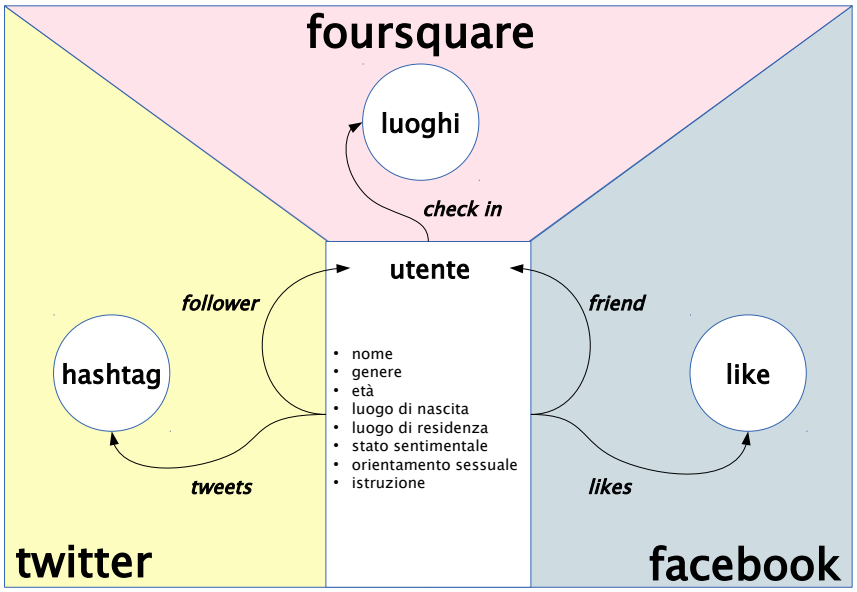


Figura 3.1: Modello dei dati

3.1.2 Problematiche di integrazione fra le sorgenti

L'obiettivo finale è definire un modello unico dei dati, come esposto in Figura 3.1, per integrare tre sorgenti eterogenee.
Una evidente differenza fra Twitter e Facebook è la natura delle interazioni fra gli utenti: Twitter realizza un modello asimmetrico di relazione, in cui la discrepanza tra il numero di persone che ci seguono e quelle che seguiamo determina la reputazione dell'individuo all'interno della comunità; viceversa in Facebook, una volta stretta amicizia, il rapporto tra due amici è paritario e simmetrico. Questa peculiarità ha sensibili implicazioni sul modello dei dati, giacché il grafo degli utenti deve adesso contenere archi orientati.
Una ulteriore peculiarità risiede nel confronto tra *like* e *hashtag*. Mentre la relazione tra un utente Facebook e un like è binaria—o l'utente ha espresso il proprio gradimento di un oggetto o non lo ha fatto—nel caso di Twitter un utente può non adoperare mai un hashtag o usarlo con frequenza. Anche in questo caso il modello dei dati richiede una estensione, tramite l'introduzione di pesi sugli archi utente-hashtag. A differenza del caso precedente, dove un arco non orientato può essere equivalentemente modellato da una coppia di archi orientati, conciliare gli archi privi di peso di Facebook con quelli pesati di Twitter richiede una normalizzazione arbitraria dei pesi. In aggiunta, il significato di un hashtag è tutt'altro che analogo ad

un like, poiché, avendo fissato l'argomento, un tweet può esprimere tanto apprezzamento quanto avversione, discussione o "inutile chiacchiera"[2]. Per amalgamare semanticamente i tweet ai like sarebbe necessario discriminare i tweet in semplici categorie—*like* e *dislike*—che tuttavia è un problema di elaborazione del linguaggio naturale e clustering in sé. Infine, la vita media di un hashtag è di gran lunga più breve rispetto ad un like. In termini assoluti, mentre un like è sostenuto da un contenuto—una pagina o una realtà esterna a Facebook—la creazione di un hashtag richiede al più uno sforzo di immaginazione; la controparte della proliferazione degli hashtag è proprio la loro volatilità. In termini relativi, il carattere episodico e conversevole del tweet richiede tempi di elaborazione ed azione più stringenti, in quanto l'associazione e il coinvolgimento dell'utente rispetto ad un hashtag può rapidamente consumarsi. In sintesi, l'analisi dei tweet richiede algoritmi tolleranti di una elevatissima dimensionalità, coerente con l'elaborazione del linguaggio naturale, possibilmente basati su grafi e che garantiscano una bassa complessità temporale anche a scapito della qualità.
In ultimo, Foursquare è un servizio unico nel suo genere e, contribuendo informazione nuova e non ridondante, non richiede sforzi addizionali di assimilazione nel modello.
Una volta definiti i criteri di integrazione, l'ultimo passo sarebbe la *entity resolution* ovvero l'identificazione dei profili che, benché apparentemente scorrelati e provenienti da sorgenti diverse, appartengono al medesimo individuo reale.
La quantità di dati a disposizione di Neosperience si è rivelata, comunque, molto al di sotto del volume prospettato a regime, sul quale gli algoritmi e le tecniche di data mining avrebbero dovuto essere messe alla prova. Pertanto, il lavoro si è articolato nei seguenti passi:

- raccolta dei dati
- selezione degli algoritmi
- preparazione dei dati
- esecuzione degli algoritmi
- analisi dei risultati

[2] http://web.archive.org/web/20110715062407/www.pearanalytics.com/blog/wp-content/uploads/2010/05/Twitter-Study-August-2009.pdf

3.2 Raccolta dei dati

Le *Graph API*[3] sono lo strumento principale che permette di leggere e scrivere sul grafo sociale di Facebook. Il grafo sociale è una rappresentazione delle informazioni presenti nel social network composto da:

- **nodi**: entità di vario tipo come utenti, foto, eventi, pagine o commenti, etichettati da un identificatore unico
- **archi**: connessioni esistenti tra le entità: relazioni d'amicizia tra utenti, foto in una pagina, commento su certo un evento, etc.
- **campi**: attributi delle entità come il compleanno di un utente o il nome di una pagina.

La maggior parte delle richieste fatte con le Graph API necessita di un *access token*[4], che consente un accesso temporaneo e sicuro alle informazioni di profilo di un utente. In particolare, l'access token deve essere composto con una serie di permessi a seconda di quale attributo si vuole leggere o scrivere: ad esempio, se si è interessati al nome, al genere e agli amici di un certo individuo, esso deve considerare, simultaneamente, i permessi `user_about_me`, `user_gender` e `user_friends`.
Per recuperare queste informazioni è sufficiente una chiamata al metodo `HTTP GET`:

```
GET https://graph.facebook.com/me?fields=name,gender,friends
```

che restituisce il seguente file JSON[5]:

```
{
  "name": "Nicola P.",
  "gender": "male",
  "friends": {
    "data": [
      { "name": "Linda",  "id": "987654321" },
      { "name": "Matteo", "id": "123123123" }
    ]
   }
}
```

3.2.1 Applicazione per la raccolta dei dati

Mediante l'SDK per PHP[6] abbiamo creato un'applicazione che ha ci permesso di recuperare da un utente le seguenti tipologie d'informazione: *profilo*

[3] https://developers.facebook.com/docs/graph-api
[4] www.oauth.org
[5] JavaScript Object Notation, http://www.json.org
[6] https://developers.facebook.com/docs/reference/php/4.0.0

(data di nascita, genere, orientamento sessuale, stato sentimentale, città natale, città di residenza e istruzione), *like* (tutte le pagine su cui si è indicato *like*) e *amici* (profilo e like degli amici).
L'applicazione è stata utilizzata da 87 utenti (che abbiamo indicato con il termine *DAU* o *Direct User Application*) ottenendo, in totale, circa 25 mila profili Facebook in formato JSON.
Il passo successivo è stato quello di unificare tutti i profili in un unico file GML[7]. Dal file JSON di ogni DAU sono stati recuperati gli identificatori degli amici e quindi i file JSON ad essi corrispondenti: in questa maniera è stato possibile costruire la rete personale del DAU (o *ego-network*) arricchita dalle informazioni di profilo e dai like di ogni nodo. La composizione incrementale di queste reti ha portato alla creazione del grafo finale.

3.2.2 Creazione dei sottografi

Come già accennato in precedenza, abbiamo sviluppato uno strumento[8] per il campionamento di grafi che, da un lato, riuscisse a creare sottografi significativi dal punto di vista topologico e degli attributi; dall'altro, che presentasse una forte componente casuale per modellare la varietà dei dati in input che potrebbero essere utilizzati successivamente.
Il procedimento adottato è il seguente:

1. si sceglie casualmente un DAU X non ancora considerato
2. si recuperano i vicini di X non ancora considerati
3. con un certa probabilità p si sceglie un vicino Y di X; altrimenti si torna a 1, scegliendo casualmente un altro DAU
4. se si è scelto il vicino Y di X si scala la probabilità p di un fattore $s < 1$, si pone $X = Y$ e si torna ad 1.

Questo procedimento viene ripetuto fino a quando non si raggiunge una certa percentuale π di utenti considerati.
Un DAU è un nodo ricco di informazioni, quindi è fondamentale che esso venga scelto come punto di partenza del campionamento. Con una probabilità che scala progressivamente, si considerano i suoi vicini e i vicini dei vicini, costruendo così una rete collegata al DAU iniziale. Successivamente,

[7] Il formato GML (Graph Modelling Language) permette la rappresentazione dettagliata di un grafo con attributi

[8] Il linguaggio utilizzato per la creazione di questo tool è Gremlin (https://github.com/tinkerpop/gremlin/wiki) che permette di comporre query su grafi

scegliendo un nuovo DAU (cioè partendo di nuovo dal punto 1) si ha la possibilità di costruire una rete ulteriore che, a seconda dei casi, sarà collegata o meno alla prima. Questo procedimento iterativo porta, infine, alla selezione di un insieme di nodi su cui è possibile indurre il grafo campionato.
Variando i tre parametri (p, s e π) siamo stati in grado di generare una vasta gamma di sottografi che presentano caratteristiche topologiche e di attributi molto variegate.

3.3 Selezione degli algoritmi

In considerazione della varietà dei dataset su cui avremmo operato, abbiamo selezionato una vasta gamma di algoritmi, cercando di differenziarli per modello dei dati utilizzato, tecnica di clustering (2.3), approccio alla dimensionalità (2.3.3) e prestazioni.
Limitatamente al modello dei dati, gli algoritmi possono essere raggruppati in tre categorie:

- clustering su attributi: NetClus [88], MOC [11], LAC [26], ORCLUS [4]
- clustering su grafo: Fast Unfolding [15]
- clustering su attributi e grafo: LatentNet [44], CESNA [96], IncCluster [99], DB-CSC [41], BAGC [95]

3.3.1 Descrizione degli algoritmi

Di seguito mostreremo, per il lettore interessato, la descrizione teorica dei vari algoritmi a nostra disposizione.

NetClus è un algoritmo per il clustering di reti composte da entità eterogenee e conformazione a stella (*star network schema*). In assenza di connessioni tra utenti, il modello dei dati descritto in Figura 3.1 riflette esattamente questo scenario: al centro della stella vi è l'utente, da cui scaturiscono archi verso gli attributi di profilo, like, hashtag e check-in, ognuno dei quali forma un nuovo tipo di entità. Il pregio di NetClus è che non si limita ad attribuire una etichetta di cluster a ciascun utente, ma produce un ordinamento dei valori di ciascuna entità in ognuno dei cluster ottenuti, semplificando notevolmente l'interpretazione dei risultati. In secondo luogo, l'approccio di NetClus riduce fortemente la dimensionalità del problema rispetto agli altri algoritmi su attributi, dove ogni nuovo like o hashtag aggiunge una dimensione

al dataset. Un ultimo punto di forza è che non richiede la definizione di una misura di similarità, ma costruisce un modello probabilistico generativo sotto l'ipotesi che la rete esibisca assortatività e *preferential attachment*, come nel caso delle reti sociali. Dato il numero di cluster, l'idea generale di NetClus si compone dei seguenti passi: generare una partizione iniziale degli oggetti ed indurre i *net-cluster* $\{C_k^0\}_{k=1}^K$ dalla rete originale sulla base di queste partizioni; costruire, per ciascun net-cluster, un modello probabilistico generativo $\{P(x|C_k^t)\}_{k=1}^K$; quindi calcolare la probabilità a posteriori $P(C_k^t|x)$ per ciascun oggetto e ricalcolare l'assegnamento degli oggetti ai cluster. I due passi precedenti sono reiterati finché i cluster non raggiungono una condizione stazionaria. Al termine, dalla probabilità a posteriori si ottiene un ordinamento degli oggetti all'interno dei cluster di appartenenza. L'algoritmo ha complessità lineare nel numero di archi, ovvero per reti sparse è approssimativamente lineare nel numero di nodi.

MOC è un algoritmo probabilistico generativo per l'identificazione di cluster non disgiunti o *overlapping*. L'algoritmo generalizza e semplifica un precedente studio sul clustering sovrapposto di espressioni genetiche [84]; con una scelta oculata del modello di probabilità e definizione di distanza—la famiglia esponenziale e la divergenza di Bregman rispettivamente—gli autori sostengono di poter applicare la tecnica a dati sparsi e con numerose dimensioni, laddove il modello gaussiano e la distanza euclidea, usati più comunemente, sono noti offrire scarsi risultati. A testimonianza della formulazione originale, i dati o matrice di espressione X sono modellati come la manifestazione di due fattori, l'appartenenza M e l'attivazione A. L'appartenenza descrive quanto i cluster nascosti nei dati si manifestano in ciascun individuo; dato che il modello è *overlapping*, ogni individuo può appartenere simultaneamente a più cluster. L'attivazione indica il peso di ciascuna dimensione dei dati in ognuno dei cluster, ossia quali caratteristiche di un individuo sono determinate più significativamente dall'appartenenza al cluster. I dati sono quindi ricostruiti come il prodotto $X' = M \times A$ tra la matrice di appartenenza e la matrice di attivazione. Seguendo questa intuizione, l'algoritmo costruisce una stima iterativa di M e A. Il vettore di appartenenza M_i di ciascun individuo è ottenuto con un approccio *greedy*, volto a minimizzare la distanza tra l'individuo e la sua approssimazione $M_i \times A$: fissata la matrice di attivazione A, MOC 'accende' con ogni iterazione un nuovo cluster, fino a quando per nessuna scelta è possibile ridurre ulteriormente la funzione obiettivo.

La complessità temporale di questa fase è $O(k^3)$ nel numero di cluster k. La matrice di attivazione può essere costruita dalla minimizzazione della medesima funzione obiettivo, fissata M; seppure sia definita una diversa formula per ogni modello nella famiglia esponenziale e relativa divergenza, la complessità temporale di questa fase può essere stimata in $O(d^3)$.

LAC è un algoritmo *centroid-based* di *subspace clustering*. Come discusso in 2.3.2, le prestazioni degli algoritmi basati sulla distanza euclidea degradano rapidamente al crescere del numero di dimensioni. Per ovviare a questo noto limite, LAC trasforma lo spazio in cui ciascun cluster è immerso, associando ad ogni cluster un vettore di pesi affinché maggiore importanza sia conferita alle dimensioni lungo le quali c'è aggregazione dei punti, all'interno del cluster. I pesi indicano quindi il grado di partecipazione di ciascun attributo al cluster; se i punti sono fortemente raggruppati rispetto a quell'attributo, il peso sarà alto, e basso altrimenti. Questi coefficienti sono appresi iterativamente con la ottimizzazione della somma degli errori quadrati (SSE), ovvero, per ciascun cluster, le distanze tra il centroide di riferimento e tutti i punti appartenenti al cluster. In assenza di contromisure, tutto il peso sarebbe concentrato sulla dimensione a minima varianza e l'algoritmo scoprirebbe unicamente cluster monodimensionali; per controllare la dimensione dello sottospazio in cui i cluster sono valutati, l'utente deve fissare un termine di penalità che è inserito nella funzione obiettivo. Sebbene l'esistenza di parametri astratti come questo sia un aspetto in genere indesiderabile, gli autori hanno definito una metodologia *ensemble* per la determinazione del valore ottimo attraverso esecuzioni multiple della procedura di clustering. Ogni iterazione dell'algoritmo ha complessità $O(kDN)$, dove k è il numero di cluster, N il numero di elementi del dataset e D il numero di dimensioni; il tempo di esecuzione è quindi governato dalla rapidità con cui l'algoritmo converge ad una soluzione.

ORCLUS è un algoritmo di clustering proiettato (*projected clustering*) basato su K-medoids. Dati due parametri specificati dall'utente, il numero di gruppi e il numero di dimensioni dei gruppi, ORCLUS identifica un insieme di cluster arbitrariamente allineati—non solo paralleli agli assi, ma ottenuti dalla combinazione lineare delle dimensioni originali—ciascuno in un sottospazio della dimensione specificata. ORCLUS è molto simile all'algoritmo k-means, tranne per il fatto che, invece di misurare le distanze nello spazio completo, queste sono cal-

colate nei sottospazi di ciascun cluster. Preliminarmente, ORCLUS sceglie casualmente dei 'semi' o potenziali medoidi. L'algoritmo richiede quindi un processo iterativo: in ogni iterazione ciascun cluster è progressivamente raffinato, rimuovendo le dimensioni prive di aggregazione e riducendo il numero di cluster dall'accorpamento di quelli più simili. I sottospazi sono formati tramite PCA, ovvero calcolando la matrice di covarianza per ciascun cluster e selezionando gli autovettori con la minore diffusione nel sottospazio del cluster, ovvero quelli associati agli autovalori più piccoli. Quando due cluster sono vicini e hanno simili direzioni degli autovettori sono fusi assieme. A chiusura di ogni iterazione è eseguita la fase di assegnamento: ogni punto è associato al seme più vicino e i semi sono ricalcolati come i centroidi dei cluster appena formati. L'algoritmo è computazionalmente intensivo $O(d^3)$ nella dimensione dei dati, in conseguenza del calcolo della matrice di covarianza e degli autovettori di ciascun cluster. Oltretutto, richiede la specificazione non solo del numero di cluster ma anche della dimensione del sottospazio in cui ciascun cluster è costruito, un parametro difficile da stimare correttamente a priori.

LatentNet è un algoritmo di clustering probabilistico, basato su un modello generativo. Partendo da un grafo di relazioni binarie, il modello assume che ogni nodo possieda una posizione non osservabile in uno spazio sociale d-dimensionale, euclideo e latente. Con questa premessa, la presenza o l'assenza di una connessione tra due individui è indipendente da ogni altra informazione, se sono note le posizioni dei due individui nello spazio sociale. Questo modello esprime in maniera innata tanto la transitività quanto l'omofilia negli attributi osservati. Per introdurre la proprietà di clustering, si assume che le posizioni nello spazio latente siano estratte da un modello misto (*mixture model*) di distribuzioni Gaussiane multivariate: ogni distribuzione è caratterizzata da una propria media e varianza, e rappresenta un diverso gruppo di individui o cluster. Per la stima delle posizioni nello spazio latente, gli autori propongono due metodi. Il primo è un metodo in due fasi: inizialmente calcola la stima a massima verosimiglianza del modello dello spazio latente—che spiega la transitività e l'omofilia—e determina le posizioni degli attori nello spazio sociale; la seconda fase stima i parametri del modello misto, basandosi sulle posizioni nello spazio latente ottenute nella prima fase. Il secondo metodo è pienamente Bayesiano e usa *MCMC sampling*: il vantaggio è che stima simultaneamente le posizioni latenti e il modello di clustering, ma è computazionalmente

più esigente.

Fast Unfolding è un algoritmo euristico per il clustering di grafi basato sull'ottimizzazione della modularità. Il metodo è diviso in due fasi che vengono ripetute ciclicamente. All'inizio, si considerano tante comunità quanti sono i nodi della rete. Successivamente, per ogni nodo i si considerano i suoi vicini j e si valuta il guadagno della modularità che si avrebbe se si spostasse il nodo dalla sua comunità alla comunità di un nodo j: il nodo i viene effettivamente spostato in quella comunità che genera il massimo guadagno positivo. Se ciò non è possibile il nodo resta nella sua comunità. Questa prima fase viene ripetuta sequenzialmente per ogni nodo, finché non è possibile nessun ulteriore miglioramento. La seconda fase consiste nel costruire un nuovo grafo i cui nodi sono le comunità trovate durante la prima fase: gli archi di ogni nuovo "nodo-comunità" sono l'unione degli archi dei nodi atomici che lo compongono. Al termine di questa seconda fase si può applicare di nuovo la prima, in un processo iterativo che termina quando non è più possibile spostare un nodo in una comunità diversa ed avere un guadagno positivo della modularità.

Inc-Cluster è un algoritmo di clustering per grafi con attributi. Per ogni coppia attributo-valore (a, v) del dataset viene aggiunto, nel grafo originale, un nodo $n_{a,v}$ che sarà collegato a tutti i vertici che presentano il valore v per l'attributo a. In questo modo viene creato un grafo *aumentato* su cui è possibile definire una funzione di distanza (la *random walk distance*) che tenga conto sia dell'informazione topologica sia del valore degli attributi dei nodi: infatti la funzione di distanza considera *simili* due individui se ci sono numerosi percorsi che li collegano. Poiché nel grafo aumentato i percorsi tra i nodi sono possibili anche grazie alla presenza degli attributi che condividono e non solo per le relazioni d'amicizia, allora è facile immaginare in che senso la funzione distanza riesce a tener conto, contemporaneamente, della ego-network dei due nodi e della similarità tra i loro attributi. L'algoritmo utilizza una tecnica molto simile a quella del K-Medoids iterando l'assegnamento di un nodo ai vari "centroidi" fino a alla convergenza della funzione obiettivo. Purtroppo il calcolo della similarità degli utenti è computazionalmente inefficiente, $O(n^{2.807})$ con n numero di nodi del grafo. Infine, Inc-Cluster richiede di specificare non solo il numero di cluster, ma anche probabilità di *restart* e la lunghezza massima del percorsi nelle random walk.

DB-CSC è un algoritmo di clustering per grafi con attributi, basato sul concetto di densità: esso individua, infatti, regioni dense sia nel grafo che nello spazio degli attributi. In particolare, per ogni nodo v vengono definiti la ϵ-*neighborhood*, cioè l'insieme dei vicini che distano al massimo ϵ da v, e la k-*neighborhood* cioè l'insieme dei vicini che raggiungono v in k passi al massimo. L'intersezione tra ϵ-*neighborhood* e k-*neighborhood* rappresenta il *vicinato locale* di un nodo. L'algoritmo definisce un cluster come un insieme di vertici O tali per cui, (1) ogni vertice $v \in O$ ha un vicinato locale molto numeroso (cioè la cui cardinalità è maggiore di $minPts$, definito dall'utente) e (2) per ogni coppia di punti in O esiste sempre un cammino che li collega. Successivamente, viene costruito un grafo *arricchito* in questa maniera: se due nodi del grafo originale hanno attributi simili e sono connessi da k archi al massimo (ovvero, se appartengono allo stesso vicinato locale) allora verranno collegati nel grafo arricchito. Dal grafo arricchito si calcolano tutti i $(minPts-1)$-*core*[9], che rappresentano delle comunità potenziali: se il grafo arricchito contiene un unico $(minPts-1)$-*core* che copre ogni nodo allora esso è un cluster valido secondo la definizione sopracitata; altrimenti bisogna ripetere questa considerazione per ognuno dei $(minPts-1)$-*core* presenti nel grafo arricchito. Ricorsivamente si individuano i grafi arricchiti indotti dai nodi di ogni $(minPts-1)$-*core* e su questi si calcolano, ancora una volta, i $(minPts-1)$-*core*: se ogni grafo arricchito è composto da un singolo $(minPts-1)$-*core* che copre tutti i suoi nodi allora esso è una comunità. Questa procedura viene, infine, generalizzata per individuare i cluster in ogni sottospazio degli attributi originali.

CESNA è un algoritmo probabilistico generativo per l'identificazione di comunità non disgiunte (*overlapping*) su grafi con attributi. Per la costruzione del modello probabilistico vengono individuate due variabili osservabili, il grafo G e l'insieme di attributi X, e una variabile latente da inferire, la funzione di *membership* F che indica, per ogni nodo, qual è il grado di appartenenza ad una certa comunità. A questo punto è possibile definire due sotto-modelli generativi. Il *modello degli archi della rete*, stabilisce qual è la probabilità che due nodi sono collegati tra loro in funzione della loro membership: essi avranno un'alta probabilità di essere collegati se condividono un elevato numero di comunità.

[9]Dato un grafo G, un c-*core* è un sottografo di G connesso e massimale in cui tutti i vertici hanno un grado maggiore o uguale a c

CESNA è l'unico algoritmo, tra quelli selezionati, che supporta i valori mancanti (*missing value*): per ogni coppia attributo-valore (a, v) si crea una *proprietà* k che viene associata a tutti e soli gli utenti che presentano il valore v per l'attributo a. Quindi, il *modello degli attributi dei nodi*, stabilisce qual è la probabilità che un certo nodo u esponga la proprietà k. Successivamente, si introduce un'altra variabile latente (anch'essa da inferire): il peso W di una proprietà all'interno di un certo cluster. Se il nodo x partecipa a un certo numero di cluster per cui il peso di k è alto, allora è molto probabile che x possieda k. Questo modello fa in modo che i nodi che appartengono alla stessa comunità condividono, probabilmente, le stesse proprietà.
Infine, viene costruito il modello *combinato* che, in estrema sintesi, rappresenta la probabilità di avere un certo grafo G e un insieme di attributi X conoscendo la funzione di membership F e il set di pesi W: $P(G, X|F, W)$. Si suppone ora che questo modello generi i dati in input all'algoritmo e quindi, per spiegarli e giustificarli al meglio, si deve determinare la coppia $\hat{F}$ e $\hat{W}$ che massimizza la verosimiglianza $logP(G, X|F, W)$. Se il grado di appartenenza di un nodo ad una certa comunità supera una soglia δ allora il nodo appartiene a quella comunità; se un nodo presenta, per tutte le comunità, una membership minore di δ allora viene considerato un outlier.
La complessità computazionale è lineare nel numero dei nodi N, degli archi E e nel numero di attributi K: $O(E + NK)$.

BAGC è un algoritmo probabilistico generativo per il clustering di grafi con attributi. Idealmente, BAGC attribuisce ad ogni possibile clustering sui dati una probabilità, cosicché il miglior clustering è semplicemente quello più verosimile, che massimizza la probabilità. Nonostante la semplicità concettuale di tale approccio, esplorare interamente lo spazio dei clustering possibili è impraticabile, e l'algoritmo propone una soluzione approssimata. Il primo passo è individuare una distribuzione parametrica che approssimi la probabilità congiunta tra il grafo considerato e i possibili clustering, modellando separatamente l'etichetta di cluster, gli attributi e gli archi per ciascun vertice. Avendo espresso il clustering come un problema di ottimizzazione—trovare il massimo a posteriori della divisione in cluster condizionatamente al grafo e agli attributi dati—devono essere caratterizzate le condizioni di stazionarietà, per potersi arrestare nella ricerca iterativa del valore ottimo. La soluzione approssimata è quindi raggiunta con un metodo variazionale.

ROCK è un tecnica gerarchica (bottom-up) per il clustering overlapping

su attributi categorici. Per raggruppare gli oggetti del dataset, l'algoritmo non utilizza una semplice funzione di similarità (come l'indice Jaccard) poiché questa può rivelarsi poco conveniente in molte situazioni. Si pensi, in maniera intuitiva, al caso in cui due punti sono *distanti* tra loro ma presentano un insieme di "vicini" molto simili: una misura standard non considererà quest'ultimo aspetto e un algoritmo gerarchico non deciderà mai di raggruppare insieme i due punti. Per ovviare a questo problema, gli autori propongono una misura *globale* che consideri non solo i punti in sé ma anche la distribuzione del loro *vicinato*. Si definisce, quindi, il numero di *link* tra due oggetti come il numero di vicini che hanno in comune;[10] inoltre, due oggetti sono detti *vicini* se la loro distanza non supera una certa soglia definita dall'utente. Come ogni algoritmo gerarchico bottom-up, ROCK inizia col considerare tanti cluster quanti sono gli oggetti del dataset per poi agglomerare, iterativamente, le coppie di cluster che massimizzano una misura di *bontà* proporzionale al numero di link. La complessità computazione è $O(m_a n^2)$ dove m_a è il numero medio dei vicini di un punto.

3.4 Preparazione dei dati

I profili utente si sono rivelati spesso inaccurati, disseminati di dati inverosimili e omissioni. Purtroppo, la cattiva qualità è una caratteristica dei dati reali, in cui il rumore, le imprecisioni della rilevazione e l'informazione mancante pregiudicano il successo del data mining. Per questo motivo, la qualità del clustering dipende tanto dalla adeguatezza della tecnica impiegata quanto dall'abilità dell'analista di preparare i dati per l'analisi.

3.4.1 Rimozione degli outlier

La ricerca degli outlier è spesso un fine in sé—come l'identificazione delle intrusioni informatiche e delle frodi bancarie—ma in questo lavoro siamo interessati unicamente a prevenire le distorsioni numeriche che un outlier produce. Identificare un individuo atipico nel contesto delle reti sociali non è semplice, poiché la numerosità delle dimensioni rende insignificanti la distanza e la densità, che sono alla base delle procedure canoniche per l'identificazione delle anomalie (sezione 2.2). Non è neppure sufficiente basarsi sulla topologia del grafo delle amicizie, e dichiarare anomali gli individui

[10] In generale il numero di link è calcolato su due insiemi di punti ed indica, semplicemente, il numero di vicini comuni a tutti gli elementi degli insiemi

debolmente connessi alla popolazione. Poiché abbiamo una visione parziale della rete, le connessioni che potrebbero contribuire ad integrare un individuo sono spesso inosservate, e questo limite si accentua ulteriormente nei campioni estratti dal dataset completo. Infine, anche i like possono essere fuorvianti, poiché risultano empiricamente molto sparsi: come discuteremo in sezione 3.4.4, la popolarità media di un singolo like è estremamente bassa, meno di 5 utenti su oltre 25000. Ci siamo pertanto limitati a ricercare gli utenti che avessero un profilo platealmente abnorme. A tal fine, abbiamo misurato tramite la distanza di Mahalanobis la dissimilarità di ciascun individuo rispetto alla media della popolazione, trasformando così un problema multivariato—in considerazione del numero di attributi di ciascun individuo—in uno univariato. A questo punto è possibile applicare al vettore di distanze una tecnica arbitraria di identificazione degli outlier: test Chi-quadro, gaussiano, Local-Outlier-Factor etc. Partendo da questa selezione, abbiamo poi ispezionato individualmente i candidati e rimosso quelli che, a nostro giudizio, fossero effettivamente incompatibili con la popolazione.

3.4.2 Inferire i valori mancanti

A ciascun utente sono associati due tipi di proprietà: il profilo e i like. I like sono ovviamente variabili da un individuo all'altro, né ha senso trattare l'assenza di un *mi piace* su un certo contenuto come valore mancante. In generale, parliamo quindi di valori mancanti solo per i dati di profilo, dove sono raccolte le informazioni biografiche dell'utente. Una ulteriore ragione per dare maggior peso all'assenza di informazione di profilo rispetto ai like è il ruolo che questi dati rivestiranno nell'analisi. Mentre i like sono fondamentali per produrre il clustering, gli attributi di profilo sono decisivi per interpretarlo. Una volta individuati i cluster, attraverso i like siamo in grado di capire quali interessi accomunano le persone; tuttavia, assumendo di non avere altre fonti di informazione, senza i dati del profilo non è possibile caratterizzare questi gruppi e descrivere le persone che ne fanno parte con le variabili della segmentazione tradizionale.
Tra gli algoritmi che abbiamo selezionato, solo CESNA [96] tollera i valori mancanti; per tutti gli altri è stato necessario produrre una versione completa dei dati. In questa fase abbiamo fatto ricorso a tre soluzioni: rimuovere l'individuo con valori mancanti (la riga del dataset), rimuovere integralmente l'attributo (la colonna del dataset) o inferire il valore laddove assente. La prima soluzione è consigliabile quando l'individuo è così povero di informazione da non produrre alcun beneficio all'analisi. La seconda soluzione, eliminare integralmente una dimensione, è necessaria quando l'attributo è

assente per una significativa porzione della popolazione: in questo scenario, rimuovere gli individui privi dell'attributo sarebbe un spreco di dati e non è nemmeno disponibile sufficiente informazione reale per inferire i valori mancanti. Dalle statistiche in Tabella 3.1 si nota che gli attributi *orientamento sessuale* e *stato sentimentale* sono specificati da una estrema minoranza di utenti e sono stati pertanto rimossi dal dataset.

genere	età	luogo di nascita	luogo di residenza	educazione	orientamento sessuale	stato sentimentale
0.78%	32.53%	28.02%	23.64%	24.29%	99.13%	99.84%

Tabella 3.1: Distribuzione dei valori mancanti per attributo

In ultimo, gli attributi non specificati possono essere dedotti tramite imputazione, ma la procedura da adottare dipende dalla natura del valore mancante. Alla radice dell'informazione incompleta vi sono cause disparate: un profilo sostanzialmente inattivo, l'irrilevanza dell'informazione per l'utente o semplicemente il desiderio di privacy. Richiamando la terminologia introdotta in sezione 2.2, inattività e irrilevanza ricadono nella tipologia MCAR, in quanto non vi è alcuna connessione tra il valore dell'attributo mancante e il fatto che non sia stato inserito. La riservatezza invece è generalmente MNAR, poiché dobbiamo conservativamente assumere che il non pubblicare una informazione possa dipendere dal valore dell'attributo per l'utente considerato. I valori mancanti nel nostro dataset sono quindi anche MNAR, e per essi non esiste una procedura univoca di inferenza. Abbiamo deciso pertanto di applicare una tecnica di imputazione ad hoc che faccia leva sull'omofilia nella rete, cioè sul fatto che gli amici di un individuo sono naturalmente selezionati tra i più simili ad esso all'interno della popolazione. Per ciascun individuo con valori mancanti, abbiamo considerato gli utenti a cui è connesso nel grafo delle amicizie e stimato da questi il valore dell'attributo mancante. A questo punto, la stima può avvenire con uno dei metodi definiti in letteratura: media, hot-deck, regressione etc. Tuttavia, quando un individuo non ha un vicinato sufficientemente ampio, o i vicini a loro volta non possiedono l'attributo mancante nel soggetto, l'imputazione non è affidabile. Pertanto, abbiamo deciso di escludere dal dataset un utente se: (1) il numero di attributi di profilo mancanti è maggiore della metà, e (2) se presenta un valore mancante per un certo attributo che, a sua volta, non è specificato da oltre la metà dei suoi vicini. Con le due fasi di rimozione dei dati impuri, la dimensione del dataset è drasticamente calata; abbiamo quantificato la perdita di informazione in Tabella 3.2.
Si noti che anche la trasformazione degli attributi può produrre valori man-

	nodi	archi *friend*	archi *like*
originale (con MV)	25853	975 382	4 391 494
completo (senza MV)	16336	710 711	3 131 019
decremento assoluto	9517	264 671	1 260 475
decremento percentuale	36.81%	27.14%	28.7%

Tabella 3.2: Perdita di informazione durante l'imputazione dei valori mancanti (MV)

canti: nel passare dai nomi dei luoghi alle coordinate geografiche, non è stato talvolta possibile identificare città e paesi inseriti dagli utenti, per i quali si è dovuto inferire un valore categorico riconoscibile e convertire quest'ultimo.

3.4.3 Selezione degli attributi

Come già anticipato nei paragrafi precedenti (sezione 2.2), le tecniche di feature selection portano dei vantaggi immediati alla cluster analysis: oltre a velocizzare gli algoritmi di mining, a ridurre la dimensione del dataset e a facilitare la presentazione dei risultati, migliorano significativamente la qualità e la granularità del clustering. Infatti, la suddivisione dei dati in gruppi omogenei è molto più semplice e nitida in uno spazio con poche dimensioni; inoltre, di frequente, i cluster emergono solamente in un sottoinsieme di attributi restando "nascosti" in uno spazio ad elevata dimensionalità (ad esempio, un gruppo di individui potrebbe essere simile per età, genere e luogo di nascita ma non per religione e città di residenza).
In questo lavoro abbiamo utilizzato il tool `FSFS`[11] che permette di identificare gli attributi più significativi all'interno di un dataset in input. È una tecnica non supervisionata che si basa sul concetto di similarità tra le feature: queste, infatti, vengono raggruppate in vari cluster omogenei per poi selezionare, da ogni cluster, l'attributo più significativo. L'algoritmo richiede all'utente di specificare il numero di proprietà h da rimuovere: è necessario, quindi, condurre degli esperimenti per la regolazione di questo parametro. Tramite la definizione di *entropia di rappresentazione* [68] è possibile capire quando un certo sottoinsieme di feature è indicativo o meno: l'entropia di rappresentazione assume valori bassi quando la maggior parte dell'informazione significativa di un dataset è contenuta lungo poche dimensioni, mentre assume valori alti quando l'informazione è equamente distribuita fra tutte le feature. Quindi, studiando l'andamento dell'entropia di rappresentazione del dataset al variare del numero di attributi selezionati, è possibile individuare il miglior valore di h.

[11] Feature Selection using Feature Similarity, `http://cse.iitkgp.ac.in/~pabitra/paper/fsfs.tar.gz`

3.4.4 Trasformazione dei dati

Come anticipato in sezione 1.1, ciascun algoritmo richiede un preciso formato degli input per poter essere applicabile, e i dati, quando non soddisfano tale formulazione, devono essere convertiti. Una frequente ragione di incompatibilità fra dati e tecniche di clustering è il tipo degli attributi—numerico o categorico—ed il nostro grafo li contiene entrambi. Anche per questo motivo, si è reso necessario scrivere dei convertitori che eseguono la trasformazione del grafo nella formulazione specifica di ciascun algoritmo. Limitatamente al tipo di dato supportato, LAC, MOC e ORCLUS sono numerici, mentre NetClus, LatentNet, CESNA, BAGC e ROCK sono categorici.

Conversione numerica dei like

Per convertire una variabile categorica in una numerica, l'approccio più consueto è generare tanti attributi booleani quanti sono i possibili valori nel dominio della variabile. A ciascun like nel dataset è stata quindi associata una variabile che, per ciascun utente, assume valore *vero* o *1* se l'utente ha espresso *mi piace* su quel contenuto e *falso* o *0* altrimenti. È chiaro che la scelta dei valori numerici corrispondenti a *vero* e *falso* è completamente arbitraria ma non ininfluente, come discuteremo nel prossimo paragrafo.

Normalizzazione

Gli algoritmi numerici, tanto quelli probabilistici quanto quelli basati sulla distanza, fanno spesso ricorso alla distanza euclidea per confrontare gli individui. Quando gli attributi possiedono intervalli di valori non comparabili—per esempio uno molto ampio, come l'età, ed uno molto compatto, come i like precedentemente trasformati—la differenza tra due individui, limitatamente al primo attributo, sarà determinante nel calcolo della distanza complessiva. Per questa ragione sono essenziali le tecniche di normalizzazione, come *min-max* e *z-score* che abbiamo utilizzato in questo lavoro. Min-Max normalizza le variabili nell'intervallo $[0, 1]$ mentre Z-score trasforma i dati affinché possiedano media nulla e varianza unitaria. Non esiste un approccio preferibile in ogni circostanza e la scelta dipende dai dati da normalizzare, ma influisce certamente sulla qualità dei risultati del clustering [51]. Tuttavia, abbiamo riscontrato che la presenza di outlier incide notevolmente nella normalizzazione min-max, che tende a comprimere i dati non anomali in un intervallo ristretto di valori. Quindi la scelta è ricaduta sulla normalizzazione *z-score*, con l'eccezione degli algoritmi (come MOC) che accettano un dataset in formato sparso. In questi ultimi, la possibilità di comprimere la

dimensione del dataset dipende dal numero di elementi nulli nella matrice dei dati, in massima parte i like, la cui trasformazione tramite *z-score* vanificherebbe la sparsità dei dati. Le tecniche di normalizzazione considerate hanno imposto la rimozione degli attributi a varianza nulla, ovvero quelli che assumono lo stesso valore sull'intera popolazione; poiché tali attributi non contribuiscono al raggruppamento, la loro eliminazione non pregiudica la qualità del clustering.

Conversione numerica dei luoghi

La trasformazione numerica applicata ai like è valida, ma ha lo svantaggio di aumentare la dimensionalità dei dati ed in alcuni casi specifici è possibile fare leva sulla semantica degli attributi per ottenere migliori risultati. Il *geocoding* è il processo che porta alla determinazione delle coordinate geografiche di un luogo—espresse in longitudine e latitudine—a partire da altre informazioni, come l'indirizzo o il codice di avviamento postale. Dal nome della città di nascita e residenza, contenuto nel profilo Facebook, siamo arrivati alle sue coordinate appoggiandoci alle *Geocoding API* di Google. A questo punto, però, è essenziale un'ulteriore trasformazione: infatti un qualsiasi algoritmo di clustering considererà le due coordinate separatamente anche se queste rappresentano uno stesso attributo. È stato, quindi, necessario far sì che la latitudine e la longitudine venissero trasformate in un unico numero (che abbiamo chiamato *ODC*, *One-Dimensional Coordinates*) preservando le distanze originali tra i luoghi. A questo proposito, abbiamo utilizzato *lo scaling multidimensionale*, ovvero una tecnica di analisi statistica che, partendo dalla matrice delle distanze tra oggetti D dimensionali, assegna ad ogni oggetto una coordinata a $\tilde{D} < D$ dimensioni, in modo tale che venga approssimata la distanza euclidea originale. Quindi, abbiamo calcolato la matrice delle distanze tra tutte le città del dataset e, tramite lo Statistics ToolBox di Matlab, ponendo $\tilde{D} = 1$ siamo riusciti ad ottenere l'ODC per ogni luogo del dataset.

Discretizzazione

La trasformazione dei like da variabile categorica a numerica ha sensibilmente cambiato la struttura del problema. Poiché il numero di like unici è di poco inferiore al milione di elementi e ciascuno di essi formerà una nuova dimensione, la complessità del problema esplode rapidamente e pochissimi algoritmi sono capaci di operare in uno spazio così sparso. Un altro dato di interesse è la popolarità dei singoli like: benché esistano oggetti largamente condivisi (oltre 200 *mi piace* sono comuni a più di 1000 utenti), in media

un singolo like è stato espresso da meno di 5 individui; pertanto il potere di aggregazione dei like in questa formulazione è prevedibilmente basso. Analogamente alla discretizzazione delle variabili numeriche, per le variabili categoriche abbiamo costruito una gerarchia concettuale, tramite la quale è possibile ridurre la dimensionalità del problema risalendo dai concetti più precisi, i like, a quelli più generali, le categorie tematiche in cui Facebook classifica i like. L'uso delle categorie è necessario per tutti gli algoritmi che non accettano e non sfruttano internamente una formulazione sparsa dei dati (ad esempio LAC). In quei casi infatti non è possibile trarre vantaggio dal gran numero di elementi nulli—i like che un utente non ha mai condiviso—all'interno del dataset, la cui dimensione su disco ed in memoria non è tollerabile. In generale, le categorie sono estremamente vantaggiose per una prima analisi dei dati, con la quale identificare approssimativamente i fattori di aggregazione degli utenti. A tal punto, le categorie più rilevanti possono essere nuovamente dettagliate nei singoli like, ed il clustering ripetuto in uno sottospazio del dataset originale a granularità più fine.
Per gli algoritmi categorici si è resa necessaria una discretizzazione delle variabili numeriche: in particolare, per l'età sono state utilizzate tecniche standard di discretizzazione *equal-width* mentre le città (identificate dall'ODC negli algoritmi numerici) sono state raggruppate per provincia grazie alle Geocoding API di Google.

3.5 Esecuzione degli algoritmi

Il primo passo dell'analisi sperimentale è stato la generazione di campioni del grafo degli utenti progressivamente più grandi: 1500, 3000, 6000 e 11000 nodi. Di ciascun campione abbiamo misurato gli indici topologici descritti in sottosezione 2.4.1, che serviranno a studiare l'effetto della struttura e del volume del grafo sulle prestazioni degli algoritmi. I grafi così costruiti sono stati dapprima sottoposti ai convertitori scritti per ciascun algoritmo, e, laddove necessario, è stata applicata la normalizzazione, la discretizzazione e l'imputazione dei valori mancanti. Infine sono stati eseguiti gli algoritmi. Descriviamo adesso per ciascuna tecnica i passi necessari a produrre i risultati sperimentali e le complicazioni che sono emerse.

LAC è una tecnica per dati numerici, e pertanto si è resa necessaria la trasformazione di ogni variabile categorica: genere, luogo di nascita e residenza, orientamento sessuale, stato sentimentale, like. Come descritto in sezione 3.4.4, per i luoghi abbiamo utilizzato le coordinate monodimensionali; per i like e per il genere, che nel nostro dataset

assume due soli valori, abbiamo invece adottato una trasformazione in attributi binari. L'algoritmo è scritto in linguaggio C e legge da file una tabella che sarà poi interamente materializzata in memoria, a prescindere dalla sparsità dei dati. Questa implementazione preclude dataset con numerose dimensioni, per cui la tecnica è limitata all'uso delle categorie di like o al più poche migliaia di attributi. Oltre ai dati, l'algoritmo richiede due parametri: il numero k di cluster nella soluzione e l'indicatore h del numero di attributi che formeranno il sottospazio di ciascun cluster. Tra i due, h è chiaramente di difficile stima, poiché influisce indirettamente sul risultato attraverso la funzione obiettivo. Un inconveniente pratico dell'algoritmo è l'impossibilità di eseguire agevolmente e automaticamente il clustering su un intervallo di valori dei parametri. A causa della rappresentazione in memoria dei dati, cambiare k o h richiede infatti la ricompilazione del programma[12].

ORCLUS similmente a LAC è un algoritmo per dati numerici, ed ha richiesto le medesime operazioni sul dataset. L'implementazione che abbiamo utilizzato per l'analisi sperimentale è contenuta nel package *orclus* per R. Anche in questo caso, la procedura deve ricevere in input una matrice densa[13]. ORCLUS richiede tre parametri all'utente: il numero k di cluster, la dimensione l del sottospazio specifico di ciascun cluster e il numero *k0* di cluster iniziali, che vengono progressivamente accorpati fino a raggiungere una soluzione della dimensione specificata k. In ogni caso, la formulazione binaria dei like genera una matrice estremamente sparsa, che risulta spesso matematicamente incompatibile con l'implementazione dell'algoritmo al crescere del parametro *k0*.

MOC come nei casi precedenti opera su dati numerici, ma, giacché realizzato in Matlab, è possibile utilizzare un formato sparso per la matrice dei dati sia su file sia in memoria. Le trasformazioni applicate sui dati sono le medesime dei due algoritmi precedenti, ma in questo caso sarebbe possibile codificare direttamente i like senza ricorrere alle categorie. L'algoritmo richiede all'utente due parametri: il numero k di

[12] Poiché in C le matrici sono allocate staticamente, variare a tempo di esecuzione k e h impone la riscrittura del codice con l'allocazione dinamica delle matrici. In alternativa, gli array di lunghezza variabile (VLA) sono stati introdotti in C99, ma alcuni compilatori di rilievo (Visual Studio) non supportano lo standard.

[13] Siamo tuttavia a conoscenza di un'altra implementazione pubblica nel framework Elki, che supporta dataset in formato sparso.
http://elki.dbs.ifi.lmu.de

cluster ed una stima iniziale dell'appartenenza di ciascun elemento ai cluster della soluzione. Questa matrice di appartenenza approssimata può essere ottenuta dall'esecuzione di un altro algoritmo di clustering, come k-means o clustering gerarchico.

LatentNet è stato distribuito dagli autori nel package *latentnet* per R. LatentNet opera su grafi con attributi e, per poter essere sottoposto all'algoritmo, il dataset deve essere convertito in un oggetto di tipo *network*. Abbiamo però incontrato da subito uno scoglio invalicabile: sebbene i modelli descritti dagli autori siano potenti, tanto l'impronta in memoria quanto il tempo di computazione richiesto per produrre una soluzione crescono brutalmente con la dimensione del campione, al punto da rendere la tecnica inutilizzabile nel nostro scenario.

NetClus esige l'esistenza di un arco tra ogni utente e ciascun tipo di entità: per gli attributi di profilo, questo si traduce nell'assenza di valori mancanti. Nonostante l'algoritmo sia calzante per il problema, l'implementazione che abbiamo ottenuto dagli autori è risultata inutilizzabile: le funzioni di ordinamento, usate nel calcolo della probabilità a posteriori del modello costruito da NetClus, sono purtroppo specifiche per il clustering di risorse bibliografiche, ed assumono un modello dei dati composto da articoli, autori, conferenze e aree di ricerca. Cambiare tali funzioni, identificare dei sostituti adeguati e scriverne il codice non avrebbe comunque garantito il raggiungimento dei risultati mostrati nell'articolo di ricerca. Pertanto abbiamo dovuto scartare NetClus.

Fast Unfolding è stato implementato del package *igraph* di R. Il metodo richiede in input un oggetto *graph* che può essere costruito a partire da un file GML; restituisce le comunità identificate e il valore della modularità del partizionamento.

CESNA fa parte della piattaforma SNAP[14] che mette a disposizione una serie di tool e librerie C++ per l'analisi e la manipolazione di reti di grandi dimensioni. Richiede in input tre file: (1) il grafo, (2) una tabella che descrive quali sono gli identificatori degli attributi di ogni utente, (3) una mappa che associa ad ogni identificatore la coppia $(attributo, valore)$. L'algoritmo permette di specificare una vasta gamma di parametri: il numero di comunità da individuare, il numero minimo e massimo di comunità nel caso si decida di identificarle automaticamente, il peso da dare agli attributi e alla rete del grafo e,

[14] http://snap.stanford.edu/

infine, il numero di thread per una versione parallela. È l'unico, tra gli algoritmi selezionati, che consente la presenza dei valori mancanti e, quindi, risulta essere efficiente anche con i singoli like di un utente. Poiché utilizza dati categorici è stato necessario discretizzare gli attributi come l'età e le città di nascita e di residenza.

BAGC similmente a CESNA opera su grafi con attributi categorici, quindi ha richiesto le medesime trasformazioni sul dataset. L'algoritmo richiede all'utente il numero di cluster da individuare e due file in input: il grafo e una matrice che specifica per ogni utente il set di attributi che lo qualifica. Poiché non sono ammessi i valori mancanti, per evitare una dimensionalità troppo elevata, è stato necessario l'utilizzo delle categorie invece dei singoli like.

DB-CSC riceve in input un grafo con attributi in formato GraphML. Richiede all'utente di impostare i parametri ϵ, k e $minPts$ descritti nella sezione 3.3. Come molti altri algoritmi, non consente la presenza di valori mancanti e, quindi, anche in questo caso è stato necessario utilizzare le categorie dei like. Purtroppo, con i dataset a disposizione, l'implementazione JAVA messa a disposizione dagli autori si è rivelata essere davvero inefficiente dal punto di vista computazionale. Per tanto DB-CSC è stato escluso dall'analisi sperimentale.

Inc-Cluster richiede all'utente di specificare il numero di cluster, la probabilità di restart e la lunghezza massima dei random walk. Come input necessita della matrice di transizione (in formato sparso) del grafo e di una tabella che specifica, per ogni individuo, gli attributi e i corrispettivi valori. L'implementazione MATLAB messa a disposizione dagli autori è sviluppata appositamente per un dataset particolare: ciò avrebbe comportato di dover reimplementare l'algoritmo e renderlo capace di accettare un qualunque input. Spinti da questa circostanza e dalle basse performance teoriche[15], abbiamo deciso di non utilizzarlo nel nostro framework.

ROCK è distribuito nel package *cba* di R; si è mostrato in generale molto efficace con i dataset a nostra disposizione, individuando cluster eccessivamente numerosi e pertanto non significativi. Per questa ragione, seppure l'algoritmo è presente nel nostro framework, si è deciso di non utilizzarlo nell'analisi sperimentale.

[15] $O(n^{2.807})$

3.6 Descrizione modulare dell'architettura

Per avere una visione d'insieme delle varie tecniche e funzionalità appena esposte, in quest'ultimo paragrafo mostriamo in maniera descrittiva quali sono i moduli teorici che compongono il nostro framework e in quale ordine sono stati utilizzati (Figura 3.2).
Il modulo `Data To Graph` permette di ottenere un unico grafo con attributi, in formato GML, partendo da un insieme di utenti Facebook, in formato JSON.
Questo grafo può essere campionato tramite il modulo `Graph Sampler` (paragrafo 3.2.2) che restituirà un insieme di sottografi con caratteristiche topologiche e di attributi molto variegate. Lo strumento può essere utilizzando anche per selezionare un gruppo di utenti che abbiano certe caratteristiche: ad esempio, si può avere la necessità di condurre una segmentazione di mercato solo sui profili Facebook che siano al di sotto di una certa età e con un particolare livello di istruzione.
Ogni grafo GML può essere sottoposto al modulo `Graph Preprocessing` (paragrafo 3.4) che è composto a sua volta dai tre strumenti per l'identificazione degli outlier, la selezione degli attributi rilevanti e l'inferenza dei valori mancanti.
Se si necessita di una valutazione topologica del grafo è allora possibile usare il modulo `Topological Metrics` (paragrafo 2.4.1): il risultato sarà composto da una serie di metriche che indicano il numero di nodi e di relazioni d'amicizia, le componenti connesse del grafo, o altre informazioni più complesse come la robustezza o l'entropia del grafo.
Il modulo `Algorithms` (paragrafo 3.5) invece, presenta, oltre a tutti gli algoritmi descritti nei paragrafi precedenti, anche un insieme di strumenti di conversione che, partendo dal grafo GML da analizzare, consente di ottenere i file di input nella rappresentazione interna di un algoritmo.
Il modulo `Communities Description` serve ad avere una rappresentazione immediata delle varie comunità: tramite un file XML, viene indicata la media, la varianza e la lista dei valori di ogni attributo del cluster ordinata per rilevanza.
Infine, il modulo `Clustering Evaluation` (le cui funzionalità verranno descritte nel prossimo capitolo insieme all'analisi sperimentale) permette di valutare la bontà di un risultato di clustering, confrontare le prestazioni dei diversi algoritmi e suggerire la tecnica migliore per un certo dataset in input.

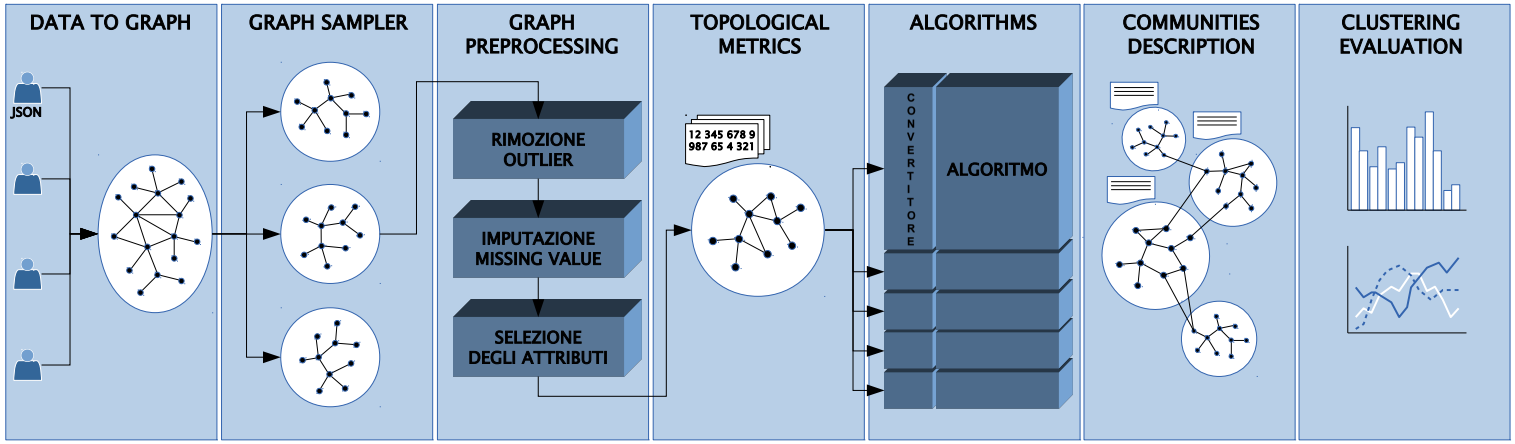


Figura 3.2: Workflow modulare dell'architettura

Capitolo 4

Analisi sperimentali

In questo capitolo descriviamo le varie funzionalità per la valutazione dei risultati di clustering. Come già anticipato, ogni algoritmo dispone di una serie di parametri in input che devono essere impostati accuratamente per elevare la qualità dei risultati. Ci proponiamo quindi di risolvere due compiti fondamentali: da un lato, tramite la misurazione degli indici di validità, stimare qual è la miglior parametrizzazione di un algoritmo su un certo dataset; dall'altro, combinare e confrontare questi risultati per ogni tipologia di algoritmo a disposizione, fornendo quindi un suggerimento su quale potrebbe essere la tecnica più efficace da utilizzare in una particolare situazione. Mostriamo, infine, le analisi sperimentali fatte sugli algoritmi che abbiamo selezionato, evidenziandone le caratteristiche salienti e i punti di forza in relazione ai dataset di test.

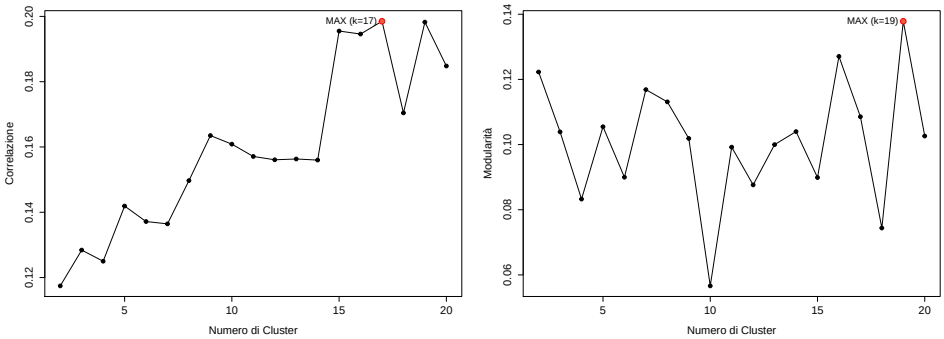
4.1 Valutazione dei risultati di clustering

L'obiettivo sostanziale di questa fase è determinare la miglior parametrizzazione di un algoritmo per ottenere soluzioni di alta qualità. In questo lavoro, abbiamo utilizzato principalmente due indici per misurare le prestazioni di una tecnica di clustering: la modularità e la correlazione. La modularità misura il grado con cui un utente è connesso agli altri membri della propria comunità rispetto a tutte le altre: varia nell'intervallo $[-1, 1]$, in cui valori prossimi ad 1 indicano che gli individui nello stesso cluster presentano una coesione molto significativa. La correlazione, invece, è calcolata tra la matrice di prossimità e la matrice di incidenza di un certo dataset e, anch'essa, varia nell'intervallo $[-1, 1]$, in cui valori vicini a 1 (-1) indicano che i punti appartenenti allo stesso cluster sono molto simili (dissimili) tra loro, mentre valori prossimi a 0 suggeriscono la presenza di un clustering casuale. Trami-

te l'utilizzo della correlazione si è liberi di impiegare una qualsiasi funzione di distanza tra gli utenti di una base di dati. In questa maniera, ad esempio, il cliente ha la possibilità di assegnare una rilevanza minore a certi attributi rispetto ad altri, mentre l'analista può scegliere una qualunque funzione che ritiene computazionalmente efficiente su un particolare dataset, senza essere vincolato da una definizione predeterminata dal framework.
Nei due paragrafi successivi descriveremo le funzionalità di impostazione dei parametri d'ingresso e gli strumenti per il confronto di due algoritmi.

4.1.1 Regolazione dei parametri d'ingresso

Tutti gli algoritmi che abbiamo selezionato permettono di specificare il numero di cluster k da individuare nel dataset in input: questo, da un lato, può essere uno svantaggio poiché è necessaria la fine regolazione di un parametro per ottenere il risultato migliore, ma dall'altro è sicuramente un beneficio in quanto lascia ampia libertà di decisione. Infatti, il numero di comunità identificate automaticamente da un algoritmo potrebbe essere poco significativo per il cliente che considera anche meccanismi esterni alla cluster analysis, ma di fondamentale importanza nella segmentazione di mercato. Questo ragionamento serve per intuire che è riduttivo identificare solamente la miglior parametrizzazione, scartando di conseguenza quelle subottimali, poiché potrebbero comunque avere un ruolo fondamentale dal punto di vista del marketing: per tale ragione, stileremo una classifica delle migliori parametrizzazioni di un algoritmo.
La Figura 4.1 mostra come variano la correlazione C e la modularità M dell'algoritmo CESNA al variare del numero di cluster k: CESNA presenta un valore massimo in $k = 17$ per C e in $k = 19$ per M. Purtroppo, se si è interessati congiuntamente alle due misure può essere difficoltoso capire per quale valore di k si ha un guadagno massimo sia dal punto di vista della correlazione che della modularità. Questo problema viene risolto costruendo uno spazio bidimensionale dove ogni punto $(c, m)_{\bar{k}}$ rappresenta la correlazione c e la modularità m misurata dopo l'esecuzione dell'algoritmo con $k = \bar{k}$. Le esecuzioni migliori sono ovviamente quelle più vicine al punto di massimo teorico $MAX = (1, 1)$. Si calcola, quindi, la distanza euclidea tra MAX e tutti gli altri punti del piano, stilando una classifica che presenta le varie parametrizzazioni dell'algoritmo in ordine crescente: quelle che occupano le prime posizioni sono le migliori in quanto più vicine al punto massimo. La Figura 4.2 esemplifica proprio questo ragionamento: ogni punto è etichettato dal valore di k e dalla posizione nella graduatoria delle distanze. Come si può notare, la parametrizzazione $k = 19$ permette di massimizzare contem-



(a) Correlazione *(b) Modularità*

Figura 4.1: Correlazione e Modularità in funzione del Numero di Cluster

poraneamente la correlazione e la modularità.
Può essere interessante, inoltre, voler assegnare un diverso peso ai due indici per specificare che uno è più importante dell'altro: di conseguenza abbiamo generalizzato la formulazione della distanza euclidea tramite una coppia di pesi a somma 1 per le due dimensioni, in modo tale che ognuna dia un contributo differente al risultato finale. Supponendo di essere in uno scenario in cui è la correlazione ad essere determinante, sempre dalla Figura 4.2 è possibile notare che la migliore soluzione resta sempre $k = 19$, seguita, questa volta, da $k = 17$ e $k = 15$.

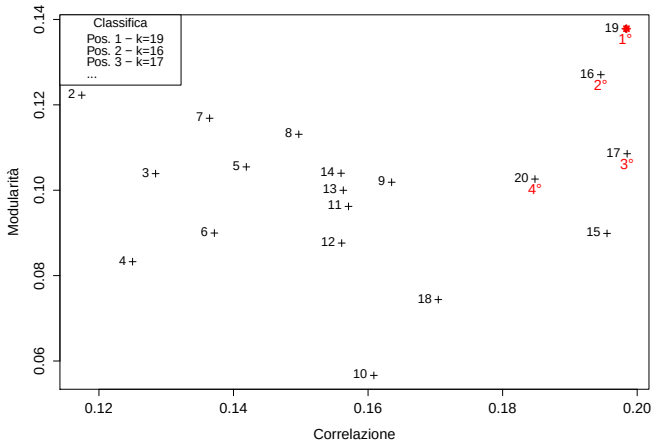


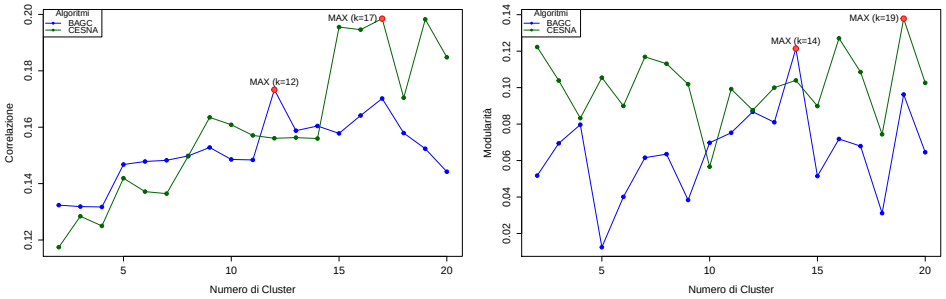
Figura 4.2: Correlazione vs Modularità

Fin qui abbiamo mostrato la situazione in cui ogni algoritmo permette di specificare un solo parametro e che siano sufficienti solo due indici di vali-

dità per le analisi dei risultati; si noti, però, che questi ragionamenti sono corretti anche in casi più complessi. Ad esempio, consideriamo un algoritmo che richiede due parametri in input e supponiamo di disporre di tre indici di bontà: per identificare la miglior parametrizzazione in funzione di un solo indice è sufficiente eseguire l'algoritmo con le diverse combinazioni di valori di ogni parametro e scegliere la coppia che massimizza l'indice di bontà. Per individuare, invece, il set di parametri che massimizza contemporaneamente tutti gli indici basta calcolare le distanze di ogni punto da $MAX = (1, 1, 1)$ per redigere la corrispondente classifica.

4.1.2 Confronto di due algoritmi

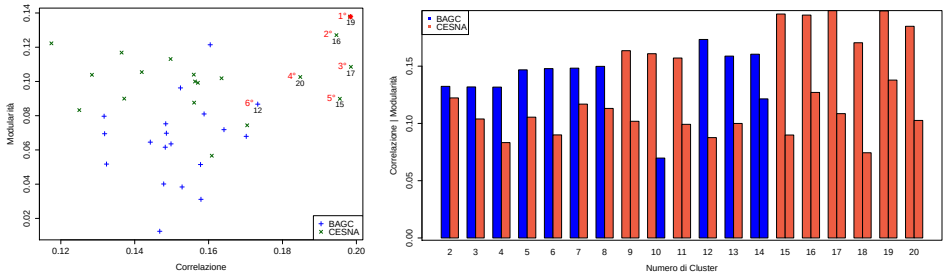
Per avere un primo confronto tra le prestazioni di due algoritmi è possibile rappresentare in un unico grafico l'andamento dei due indici di validità al variare del numero di cluster su uno stesso dataset: ad esempio, la Figura 4.3a confronta la correlazione degli algoritmi CESNA e BAGC. Come è facile intuire, BAGC raggiunge un picco $bagc_{max}$ in $k = 12$ mentre CESNA arriva al valore massimo $cesna_{max}$ in $k = 17$: se si è interessati all'algoritmo che presenta la miglior correlazione in assoluto, allora la scelta ricade, ovviamente, su CESNA con la parametrizzazione $k = 17$, poiché $cesna_{max} > bagc_{max}$. Questa analisi, come si può immaginare, è troppo semplicistica per due ragioni. Da un lato, è sempre necessario tener conto delle esecuzioni non ottimali dell'algoritmo, in quanto, relativamente al parametro k, il cliente potrebbe essere interessato ad un numero di comunità diverso da quello proposto dall'algoritmo stesso. Inoltre, quando la correlazione o la formulazione della misura di distanza tra gli individui non sono sufficienti ad identificare risultati significativi, è utile analizzare le soluzioni subottime che potrebbero essere proprio quelle che presentano informazioni essenziali per il cliente.
La situazione si complica ulteriormente quando si devono considerare due indici di validità (ad esempio, oltre alla correlazione, anche la modularità): in questo caso è indispensabile studiare il *trade-off* fra di essi, ed individuare le parametrizzazioni che, da un lato, soddisfino al meglio gli interessi del cliente e, dall'altro, presentino dei valori importanti di correlazione e modularità. Quindi, abbiamo pensato di mostrare nel consueto piano bidimensionale i punti $(c, m)_{a,\bar{k}}$ che rappresentano la correlazione c la modularità m quando si esegue l'algoritmo a impostando $k = \bar{k}$. In questo modo è possibile identificare l'algoritmo che, con la propria parametrizzazione, offre il miglior risultato: come si nota dalla Figura 4.4a, è CESNA ad avere prestazioni superiori proprio perché le sue esecuzioni sono le più vicine al massimo teorico $(1, 1)$. Da questa rappresentazione grafica non è però immediato identificare,



(a) Correlazione *(b) Modularità*

Figura 4.3: Correlazione e Modularità in funzione del Numero di Cluster

ad esempio, un intervallo di valori di k in cui un algoritmo ha prestazioni migliori su un certo indice di validità. Per ovviare a questa limitazione produciamo un ulteriore grafico rappresentato dalla Figura 4.4b: ogni algoritmo è codificato da un colore diverso e, per ogni k, vengono presentate due barre affiancate che rappresentano, rispettivamente, la correlazione (a sinistra) e la modularità (a destra) dell'algoritmo migliore: ad esempio, per $k = 2$ BAGC presenta prestazioni superiori sulla correlazione e CESNA sulla modularità, mentre per $k = 9$ è CESNA a raggiungere il miglior risultato su entrambe le misure.



(a) Correlazione vs Modularità *(b) CESNA vs BAGC*

Figura 4.4: Correlazione e Modularità a confronto in due algoritmi

4.2 Studio dei risultati sperimentali

Nel seguito discuteremo i risultati sperimentali degli algoritmi selezionati, evidenziandone le caratteristiche salienti nei diversi scenari d'applicazione.

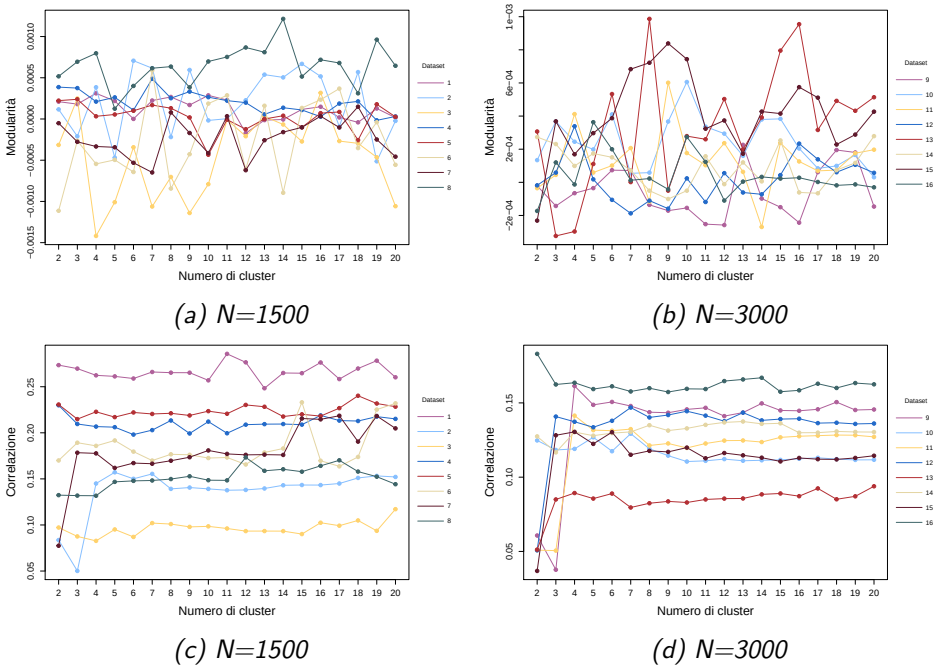
4.2.1 CESNA

CESNA è un algoritmo probabilistico per il clustering overlapping di grafi con attributi che richiede in input il numero di cluster k della soluzione.

Qualità del clustering

Per giudicare la qualità del clustering, in CESNA bisogna tener conto della modularità e dell'indice di correlazione tra la matrice di incidenza e prossimità. Dalla modularità, come si nota nella Figura 4.5a e Figura 4.5b, è praticamente impossibile individuare un *trend* che possa dare informazioni utili per un utilizzo corretto dell'algoritmo: infatti la correlazione media calcolata sui vari vettori di modularità è prossima a zero (dell'ordine di 1×10^{-4}) indicando una completa mancanza di regolarità tra le diverse rilevazioni.
Per quanto riguarda invece l'indice di correlazione (Figura 4.5c e Figura 4.5d) è possibile notare una lieve tendenza comune nei vari risultati sperimentali:

- in generale, i picchi massimi di correlazione si presentano per alti valori di k; spesso si nota come bassi valori di k corrispondono ad una correlazione molto ridotta



(a) N=1500 (b) N=3000

(c) N=1500 (d) N=3000

Figura 4.5: Valori di modularità (a,b) e correlazione (c,d) al variare del parametro k

- la correlazione risulta avere un andamento relativamente stabile al variare di k e mostra una decrescita media all'aumentare del volume del dataset

Confrontato con gli altri algoritmi, CESNA presenta solitamente valori di correlazione più bassi. La motivazione è da ricercare nel fatto che l'algoritmo cerca di massimizzare la qualità del clustering sia dal punto di vista degli attributi che da quello topologico: tener conto di due diversi fattori porta inevitabilmente alla ricerca di un trade-off tra di essi che può penalizzarli entrambi in virtù di una ottimizzazione congiunta.

Tempo di esecuzione

La Figura 4.6 riporta i tempi medi d'esecuzione per i campioni del grafo originale: notiamo come questi siano abbastanza prevedibili in quanto crescono non solo all'aumentare del numero di cluster k su ogni dataset, ma anche all'aumentare della dimensione dei grafi.
Come si osserva, CESNA è un algoritmo piuttosto veloce: sui dataset molto semplici è al di sotto di 2 minuti d'esecuzione, mentre su grafi più articolati (come quelli da 11000 nodi) presenta un tempo medio non superiore agli 8 minuti. C'è da precisare, però, che a differenza degli altri algoritmi l'implementazione di CESNA che disponiamo è sviluppata con 4 thread che lavorano in parallelo.

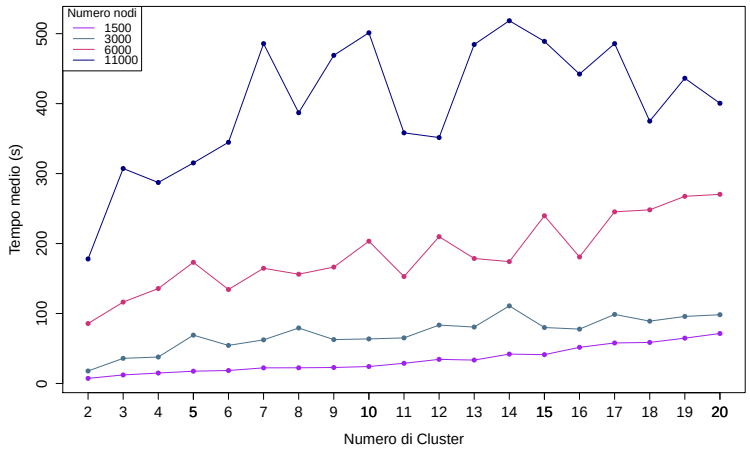


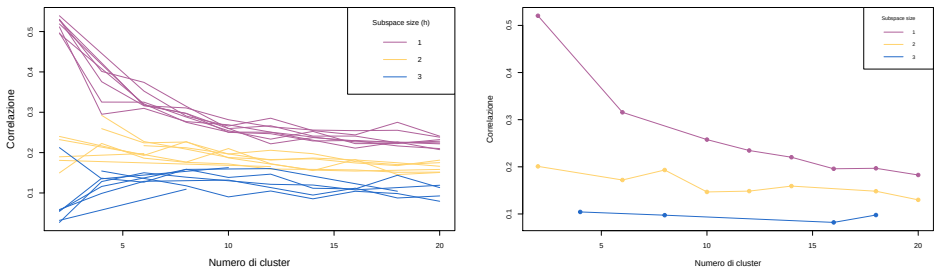
Figura 4.6: Tempi di esecuzione medi

4.2.2 LAC

LAC è un algoritmo di subspace clustering su attributi basato sulla distanza, che deve essere configurato con il numero di cluster k della soluzione ed un indicatore h del numero di dimensioni in ciascun sottospazio.

Qualità del clustering

Come mostrato in Figura 4.7, entrambi i parametri dell'algoritmo ne influenzano visibilmente il risultato. È interessante osservare che campioni significativamente diversi estratti dal grafo completo mostrano la stessa evoluzione della prestazione al variare di k; tuttavia, al crescere di h la rilevanza del numero di gruppi cala costantemente, fino a divenire insignificante sulla qualità del clustering per $h \geq 3$. Appare specialmente controintuitivo che la qualità della soluzione peggiori al crescere di h, man mano che i sottospazi si avvicinano allo spazio completo degli attributi. Ricordiamo infatti che, per il calcolo della correlazione, la similarità tra gli individui è calcolata nello spazio completo, per cui un incremento di h dovrebbe avvicinare le distanze calcolate dall'algoritmo a quelle considerate dalla correlazione. Questa tendenza—la diminuzione della qualità al crescere di h—si conserva al variare della cardinalità e del numero di attributi nel dataset. Abbiamo quindi analizzato come cambia la qualità del risultato quando si proiettano i cluster nel sottospazio in cui sono stati formati. Essendo una tecnica di subspace clustering, LAC genera ciascun gruppo in uno spazio trasformato, dando un peso trascurabile alle dimensioni prive di aggregazione, ed il numero di dimensioni caratteristiche di ciascun sottospazio è governato da h. È stata quindi calcolata la similarità tra gli individui nello spazio originale e successivamente in quello trasformato dal clustering, e valutata la correlazione con la matrice di incidenza in entrambi i casi. Sebbene gli individui sarebbero



(a) N=1500 (b) N=3000

Figura 4.7: Correlazione al variare del parametro h

dovuti apparire più simili nello spazio trasformato, la correlazione invece decresce leggermente, come si può osservare in Figura 4.8. Questi risultati mettono in luce un limite alla capacità di effettuare raccomandazioni: tanto più la percezione della similarità tra gli individui è diversa fra l'analista e l'algoritmo, quanto più è difficile stimare a priori l'efficacia dell'algoritmo su dati nuovi.

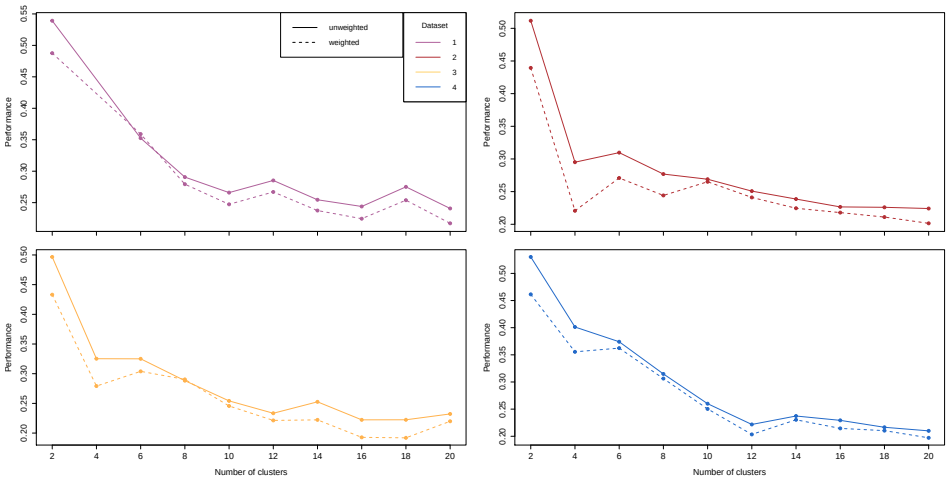
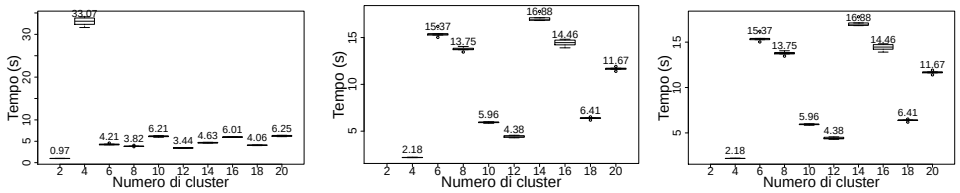


Figura 4.8: Prestazioni di LAC sui cluster pesati

Tempo di esecuzione

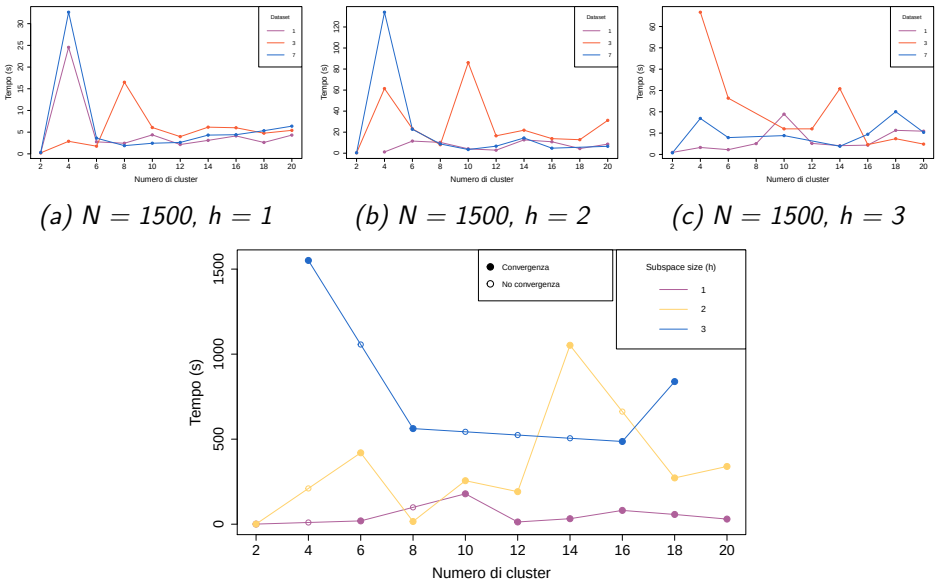
LAC mostra dei tempi di computazione impredicibili, ma fortemente connessi tanto ai valori scelti per i parametri quanto ai dati sotto esame. L'aspetto non banale di questa dipendenza è che non altera significativamente il tempo di esecuzione di una singola iterazione dell'algoritmo—che ricordiamo essere $O(kDN)$—ma incide con grande variabilità sul numero di iterazioni prima della convergenza. Esaminando Figura 4.10a si nota che al variare del dataset, avendo fissato il numero di cluster e la dimensione del sottospazio, i tempi di esecuzione fluttuano largamente, così come sullo stesso dataset è sufficiente variare k o h per rallentare o accelerare la convergenza. Sui tempi di esecuzione non sembrano invece avere rilievo le condizioni iniziali dell'algoritmo, quali ad esempio la scelta dei centroidi, come si nota in Figura 4.9. Fissato il parametro h, ripetute esecuzioni dell'algoritmo per un dato k hanno prodotto tempi di esecuzione pressoché stazionari, né le oscillazioni sono dovute ad un diverso ritmo di convergenza, in quanto il numero di iterazioni è risultato costante fra le diverse rilevazioni. Benché sia difficile notare un chiaro andamento dei tempi di esecuzione in funzione degli input, possono tuttavia essere avanzate alcune considerazioni. Dagli esperimenti



(a) $h = 1$ *(b) $h = 2$* *(c) $h = 3$*

Figura 4.9: Variabilità dei tempi di esecuzione di LAC sul dataset 1

risulta che LAC non raggiunge facilmente la convergenza per alcuni valori del parametro k in relazione al dataset sotto analisi. Al riguardo, si noti il tempo richiesto per calcolare il clustering di un dataset di 3000 nodi per $k = 4$ e $h = 3$ (Figura 4.10d) rispetto agli altri valori del numero di cluster; oltretutto, con lo stesso k e per $h = \{1, 2\}$ l'algoritmo non è arrivato a convergenza in tempi congrui con la dimensione del campione. Questi picchi peggiorano significativamente all'aumentare di h e della dimensione del dataset.



(a) $N = 1500$, $h = 1$ *(b) $N = 1500$, $h = 2$* *(c) $N = 1500$, $h = 3$*

(d) $N = 3000$

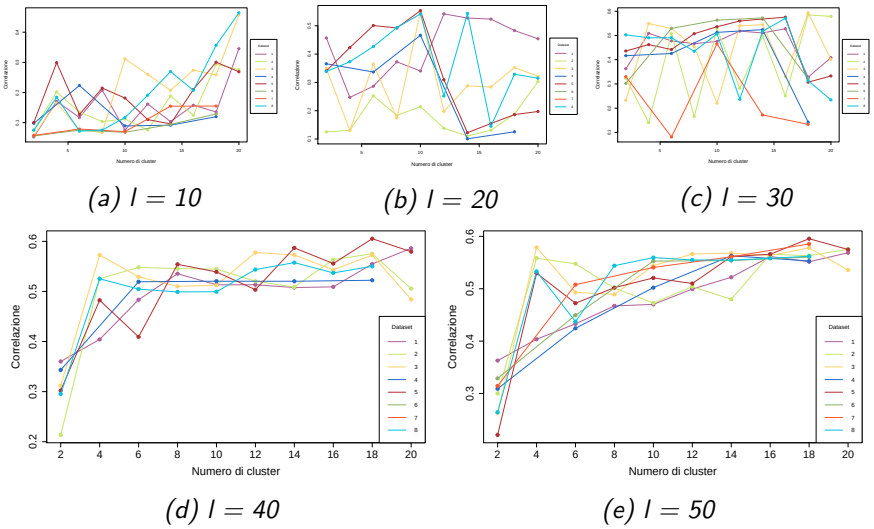
Figura 4.10: Tempi di esecuzione di LAC. Nelle parametrizzazioni contrassegnate da no convergenza l'algoritmo non ha prodotto un risultato dopo diverse ore di elaborazione

4.2.3 ORCLUS

ORCLUS è anch'esso un algoritmo di subspace clustering su attributi basato sulla distanza, che richiede il numero di cluster k della soluzione, il numero k_0 di centroidi iniziali e la dimensione l del sottospazio.

Qualità del clustering

In Figura 4.11 è riportato l'andamento della correlazione su diversi dataset al variare del parametro l. Sebbene per valori intermedi ($l = 20$, $l = 30$) la prestazione segua un corso imprevedibile, è invece interessante osservare come per valori molto bassi ($l = 10$) e relativamente alti ($l = 40$, $l = 50$) la correlazione sia invece regolare. Per comodità, mostriamo in Figura 4.12b l'andamento medio della qualità del clustering, che ripetiamo è realmente indicativo solo per $l \in \{10, 40, 50\}$. Dalla figura emergono due osservazioni: la prima è che più il sottospazio è piccolo rispetto alla dimensionalità originale dei dati e più diviene irrilevante il confronto con algoritmi che non praticano subspace clustering, i quali considerano tutte le proprietà di un individuo nella valutazione della similarità. La seconda è che, oltre una certa grandezza del sottospazio, aggiungere ulteriori dimensioni non è vantaggioso, ed appare palesemente dal confronto tra $l = 40$ e $l = 50$.
La manifesta instabilità delle prestazioni in alcuni dataset, come si nota in Figura 4.12a per $l = 20$ (giallo) e $l = 30$ (blu), ci ha spinto a studiare la robustezza delle prestazioni di ORCLUS, ovvero la capacità dell'algoritmo di correggere in itinere le decisioni iniziali, come la scelta dei medoidi, e la



(a) $l = 10$ (b) $l = 20$ (c) $l = 30$

(d) $l = 40$ (e) $l = 50$

Figura 4.11: Correlazione al variare del parametro l

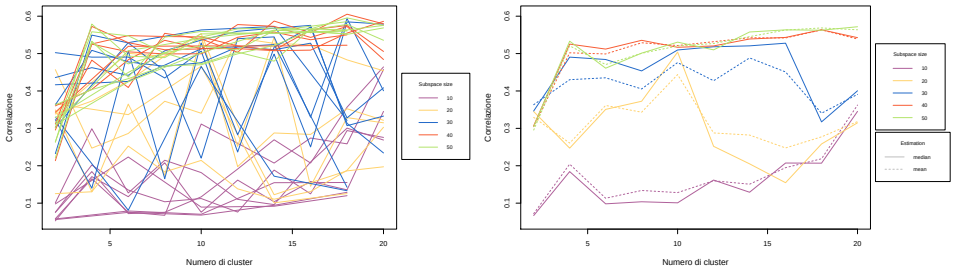


Figura 4.12: Prestazioni di ORCLUS su campioni di 1500 nodi

formazione del sottospazio di ciascun cluster.
Abbiamo pertanto reiterato il clustering sullo stesso dataset, variando k e l, e misurato la variazione del punteggio tra le iterazioni. I risultati sono riportati in Figura 4.13: come si può notare, per certune parametrizzazioni dell'algoritmo i risultati sono compresi in una ampia forbice di valori. Una concausa di questo fenomeno è che quando l'algoritmo è costretto a forzare un cluster ben definito in un sottospazio più piccolo, le l dimensioni sono scelte casualmente, portando a risultati diversi pur partendo dalle stesse condizioni iniziali. Poiché anche ORCLUS è un algoritmo di subspace clustering, abbiamo studiato quanto siano coesi i cluster proiettati nel proprio sottospazio. In Figura 4.14 abbiamo riportato le analisi svolte su dataset di 1500 e 3000 nodi, per $l \in \{10, 30, 50\}$, dove la prestazione originale è riportata con una linea continua e quella proiettata con una linea tratteggiata. È significativo che per $l = 10$ la qualità dei cluster proiettati è estremamente elevata, e questo comportamento è confermato in tutti i dataset considerati; ciò contribuirebbe inoltre a spiegare la corrispondente bassa qualità dei cluster quando sono valutati nello spazio completo degli attributi. L'alto

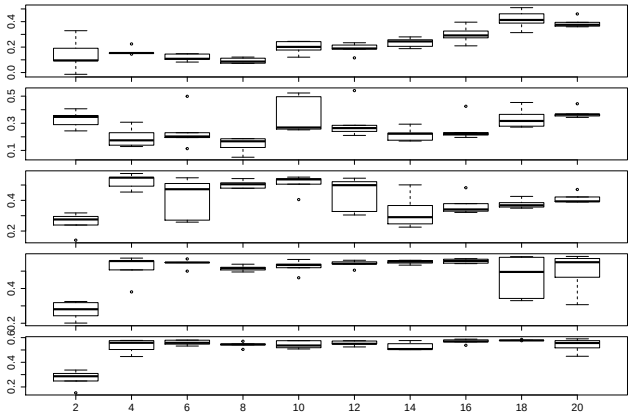
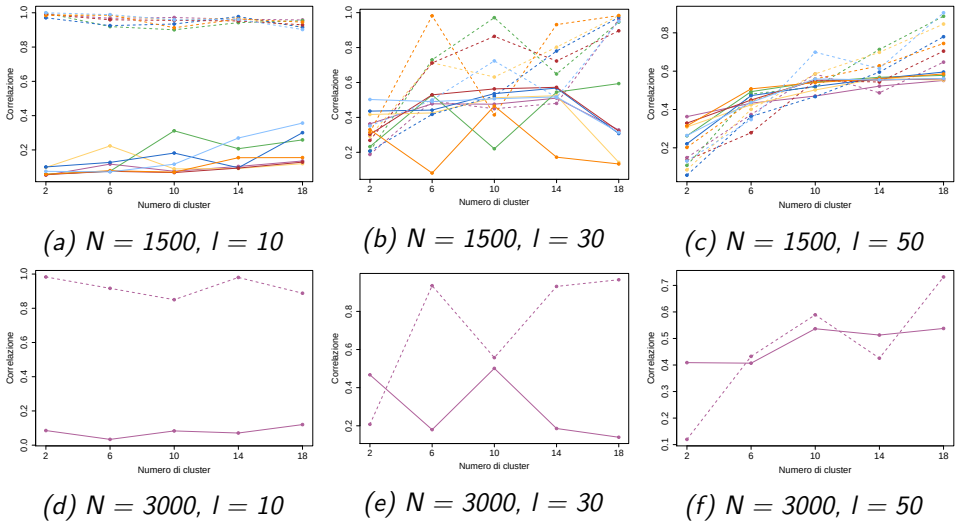


Figura 4.13: Instabilità delle prestazioni, partendo dall'alto per $l \in \{10, 20, 30, 40, 50\}$



(a) $N = 1500$, $l = 10$ (b) $N = 1500$, $l = 30$ (c) $N = 1500$, $l = 50$

(d) $N = 3000$, $l = 10$ (e) $N = 3000$, $l = 30$ (f) $N = 3000$, $l = 50$

Figura 4.14: Prestazioni di ORCLUS sui cluster proiettati

punteggio del clustering proiettato è tuttavia ingannevole, in quanto per piccoli valori di h ORCLUS forma cluster molto sbilanciati, concentrando la popolazione in un singolo gruppo ed offrendo in genere una soluzione non significativa. Per $h = 30$ non esiste una chiara tendenza: seppure i cluster proiettati siano mediamente più puri di quelli calcolati sui dati originali, le peculiarità di ciascun dataset sembrano essere dominanti. Tuttavia inizia a mostrarsi un andamento comune che si rafforza al crescere della dimensione del sottospazio ($h = 50$), per cui la qualità delle proiezioni peggiora per bassi valori di k; le stesse tendenze sono confermate all'aumentare del volume dei dati.

Tempo di esecuzione

ORCLUS mostra tempi di esecuzione indipendenti dalla dimensione l del sottospazio considerato, come mostrato in Figura 4.15. Le sottili variazioni presenti nel grafico riflettono esattamente le lievi differenze nella cardinalità dei dataset utilizzati per questa analisi (Tabella A.1). Sebbene l sia ininfluente sulla complessità temporale, il ruolo di k_0 è invece significativo. Ricordiamo che ORCLUS procede da una stima iniziale di k_0 gruppi che vengono poi accorpati fino al raggiungimento di una soluzione della dimensione k desiderata. In tutti gli esperimenti realizzati, il tempo di esecuzione è lineare in k fino ad un certo valore, dopo il quale inizia addirittura a decrescere, come mostrato in Figura 4.15. Ciò è dovuto al fatto che, per $k \leq 14$, k e k_0 sono stati fatti crescere linearmente; abbiamo poi limitato k_0 ad un

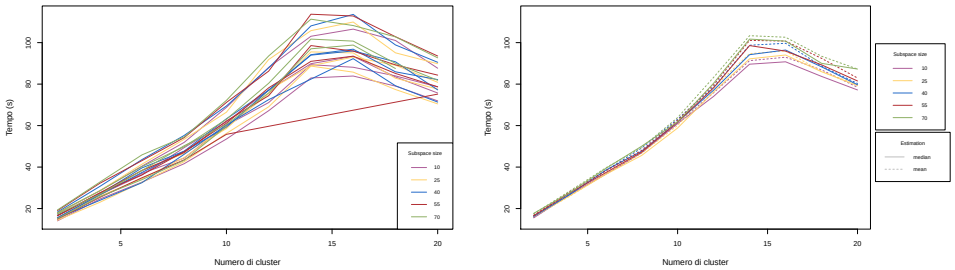


Figura 4.15: Tempi di esecuzione di ORCLUS (1500 nodi)

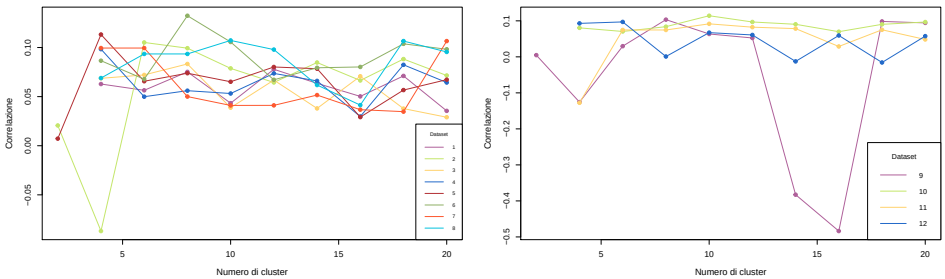
valore massimo e continuato ad incrementare k. Questo ha come conseguenza che l'algoritmo richiede meno iterazioni per raggiungere la convergenza, poiché il numero di cluster da accorpare per passare da k_0 a k decresce al crescere di k, spiegando la flessione dei tempi all'aumentare della dimensione della soluzione. È importante tenere presente che agendo sulla grandezza di k_0 rispetto a k si può governare tanto la complessità temporale dell'algoritmo quanto la affidabilità e la ripetibilità della soluzione, che entrambe crescono con k_0.

4.2.4 MOC

MOC è un algoritmo probabilistico per il clustering *overlapping* su attributi, che richiede unicamente il numero k di cluster nella soluzione.

Qualità del clustering

La Figura 4.16 mostra i risultati dell'analisi sperimentale eseguita su MOC. Sebbene non esista una tendenza sistematica delle prestazioni al variare di k, la correlazione che abbiamo registrato è stata sempre bassa, e talvolta



(a) N = 1500 *(b) N = 3000*

Figura 4.16: Prestazioni di MOC

negativa. Questi valori numerici, al confine del clustering casuale, sono spiegati dalla distribuzione dei nodi nei cluster. MOC tende a formare cluster estremamente sbilanciati, di cui la maggior parte sono vuoti e pochissimi, talvolta uno solo, raccolgono quasi l'intera popolazione. Questo comportamento è descritto in Figura 4.17: per ciascun valore di k abbiamo misurato la distribuzione del numero di nodi per cluster, ottenuta dal clustering di 8 dataset di 1500 nodi ciascuno. Se i nodi fossero perfettamente distribuiti tra i k cluster osserveremmo una bassissima varianza e una media non nulla pari a N/k. In realtà, nelle soluzioni con $k \leq 4$ il numero di cluster pieni (solitamente uno solo) è comparabile alla quantità di cluster sostanzialmente vuoti, producendo una elevatissima varianza della dimensione dei gruppi; al crescere di k, i cluster vuoti diventano predominanti e la media e varianza si azzerano, relegando come outlier i cluster non vuoti, che contengono effettivamente la popolazione. Poiché la maggior parte degli individui è compressa in un solo cluster, la correlazione tra la similarità e l'appartenenza allo stesso cluster precipita. Questo comportamento sembra essere peculiare dell'algoritmo, e non dipende né dalla dimensione del dataset né dal numero di attributi né dalla formulazione dei dati (ad esempio la conversione numerica dei like).

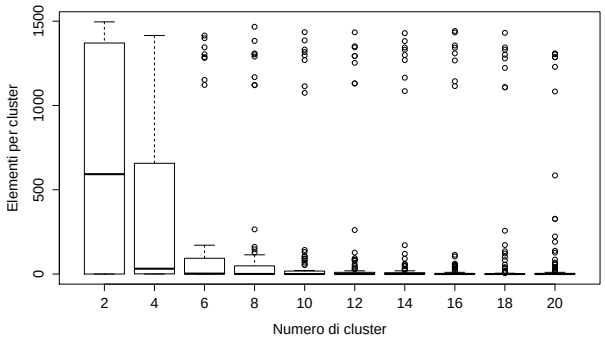
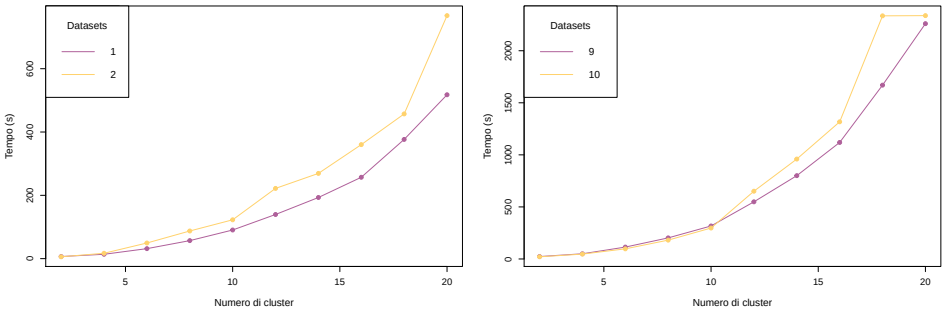


Figura 4.17: Asimmetria dei cluster

Tempo di esecuzione

Fissato N, la complessità temporale di MOC è determinata dal contributo delle due fasi dell'algoritmo: la prima è $O(k^3)$, con k il numero di cluster, e la seconda $O(d^3)$, dove d è il numero di attributi nel dataset. Il primo contributo è chiaramente visibile in Figura 4.18. La complessità della seconda fase ha invece reso impraticabile il clustering attraverso i singoli like, sebbene l'algoritmo ammetta un formato sparso per i dati; farlo avrebbe accresciuto il numero di dimensioni ben oltre la cardinalità del dataset, por-

tando di fatto ad un algoritmo di complessità cubica. Ciononostante, anche usando le categorie i tempi assoluti di computazione sono superiori agli altri algoritmi su attributi che abbiamo considerato, e non appaiono neppure giustificati dalla qualità del risultato.



(a) N = 1500 *(b) N = 3000*

Figura 4.18: Tempi di esecuzione di MOC

4.2.5 Analisi comparativa: LAC e ORCLUS

Abbiamo considerato i migliori algoritmi su attributi fra quelli selezionati per mostrare come sia possibile stimare numericamente la qualità della soluzione, per decidere quali clustering meritino di essere analizzati nel dettaglio. Dallo studio delle prestazioni sui dataset di test sono emerse delle peculiarità di ciascun algoritmo che possono aiutare nella selezione di una buona tecnica per nuovi dati in ingresso. In particolare, LAC funziona bene per popolazioni di piccole dimensioni nelle quali si vogliono identificare poche comunità in sottospazi ridotti ($h = 1$). Viceversa, ORCLUS presenta dei limiti nei piccoli dataset e offre risultati fuorvianti quando le dimensioni selezionate sono poche. Questa differenza emerge in Figura 4.19, dove abbiamo riportato al variare della dimensione del clustering le tre migliori parametrizzazioni per ciascun algoritmo. La complessità di questo confronto è duplice: da un lato entrambi gli algoritmi richiedono molteplici parametri, che non possono essere messi facilmente in corrispondenza e che, come abbiamo visto, incidono significativamente sul risultato; dall'altro, sono entrambi algoritmi di subspace clustering e quindi, a parità di numero di cluster k, i clustering di LAC e ORCLUS descrivono comunque due realtà diverse e non paragonabili[1]: non solo si differenziano per gli individui che confluiscono in ciascun

[1]Le tecniche convenzionali di confronto tra clustering tengono unicamente conto dell'assegnazione dei punti ai cluster, ignorando la somiglianza o diversità dei sottospazi associati. In particolare, LAC conserva in genere la dimensionalità originale, stabilendo però una graduatoria degli attributi tramite pesi; ORCLUS viceversa riduce effettivamente la dimensionalità dei singoli cluster combinando linearmente gli attributi originali.

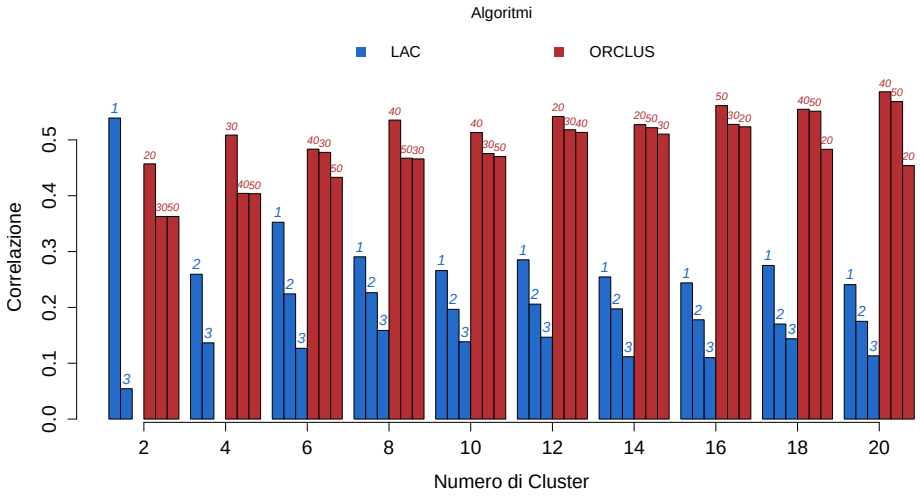


Figura 4.19: Migliori risultati di LAC e ORCLUS al variare della dimensione del clustering

cluster ma anche per gli attributi che rendono simili gli individui all'interno dello stesso gruppo. Pertanto, con questo tipo di algoritmi è imprescindibile una valutazione caso per caso della rilevanza del sottospazio in cui giace un cluster: se gli attributi selezionati per formare la comunità sono irrilevanti per l'analista, la soluzione sarà comunque da scartare a prescindere dal punteggio ottenuto.
Una metrica totalmente ortogonale alla qualità del clustering è la complessità temporale. L'ottimizzazione simultanea di due misure di qualità ha raramente un punto ottimo globale, ma richiede un compromesso soggettivo; nel formulare questa decisione è utile considerare i risultati sperimentali esposti in Figura 4.20. Come si può osservare, LAC richiede tempi sostanzialmente inferiori a ORCLUS per produrre una soluzione, seppure in genere la qualità del risultato sia anch'essa minore. Abbiamo quindi tracciato il fronte di Pareto[2] nel grafico Correlazione–Tempo e identificato i punti di maggior interesse. Come è emerso dalle analisi svolte sui dataset di test, LAC mostra una buona prestazione complessiva per $h = 1$ e $k \leq 4$ mentre ORCLUS mantiene risultati di buon livello al crescere di k.
Nella scelta del numero di comunità da identificare nella popolazione, il parametro k può essere fissato a priori dall'analista, se ha delle informazioni esterne sulla struttura del mercato o cerca di verificare nei dati un modello teorico, oppure può essere determinato a posteriori come il valore che

[2]Il fronte di Pareto è un insieme di soluzioni nell'ottimizzazione multiobiettivo; è costituito da tutti i punti non dominati, tali per cui non esiste alcun altra soluzione che sia migliore di essi per tutti gli obiettivi considerati nella funzione di ottimizzazione.

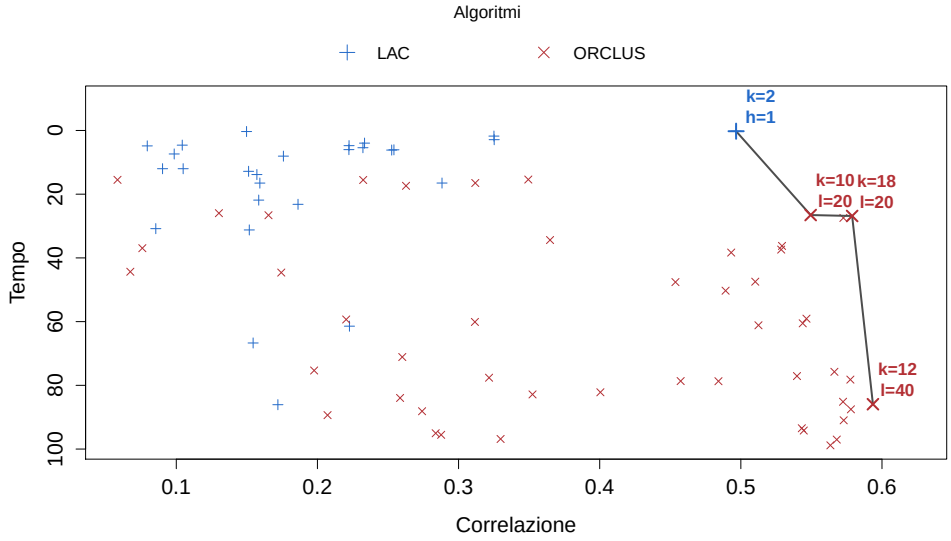
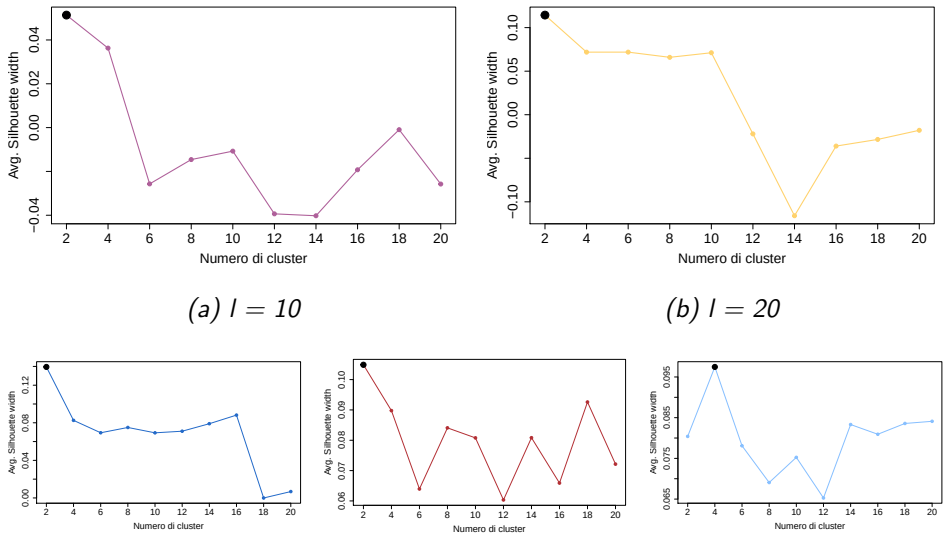


Figura 4.20: Correlazione vs Tempo di esecuzione per LAC e ORCLUS

massimizza la qualità percepita del clustering. In quest'ultimo caso, l'indice Silhouette è di grande aiuto. A differenza della correlazione, l'indice Silhouette assegna un punteggio ad un clustering che misura tanto la purezza, ovvero la similarità intra-cluster, quanto la sintesi, ossia la dissimilarità inter-cluster. Calcolando l'indice per ogni k plausibile si ottiene un grafico come quello in Figura 4.21, il cui massimo indica il k più appropriato



(a) l = 10

(b) l = 20

(c) l = 30

(d) l = 40

(e) l = 50

Figura 4.21: Indice Silhouette al variare della dimensione del sottospazio

per l'algoritmo considerato sui dati sotto esame. Una volta identificato il $\bar{k}$ ottimale, è comunque utile studiare quanto ciascun gruppo nella partizione corrente sia ben formato; a tal fine, abbiamo mostrato in Figura 4.22 la distribuzione del coefficiente Silhouette elemento per elemento, in ciascuno dei $\bar{k}$ cluster prodotti dall'algoritmo. Come si nota, l'indice sull'intera soluzione è ottenuto dalla media di contributi eterogenei, ed è necessario approfondire l'analisi per rilevare gruppi con pochi utenti e bassa qualità che potrebbero essere vantaggiosamente rimossi dal clustering. A questo proposito, si prenda ad esempio il cluster *4* in Figura 4.22, che contiene pochi individui disomogenei, il cui indice Silhouette è in genere nullo (individui distanti dalla propria comunità) o perfino negativo (individui che avrebbero dovuto essere classificati in altri cluster).

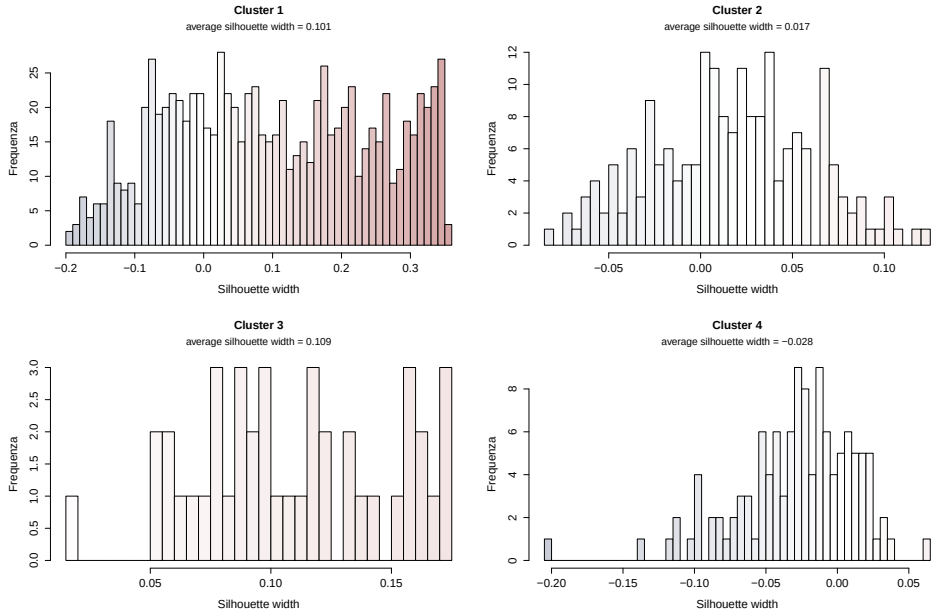


Figura 4.22: Indice Silhouette per ciascun cluster

Capitolo 5

Conclusioni e sviluppi futuri

5.1 Conclusioni

Questo lavoro è stato motivato dall'esigenza di Neosperience—una azienda che sviluppa strumenti software per la *Digital Customer Experience*—di avere uno strumento per poter valorizzare i dati degli utenti raccolti tramite la piattaforma Neosperience: in particolare, l'obiettivo principale è quello di fornire delle tecniche di data preprocessing e degli algoritmi di clustering che siano di supporto alla segmentazione di mercato. La segmentazione di mercato è definita come quel processo che permette di partizionare un ampio insieme di clienti in piccoli gruppi (o *segmenti*) caratterizzati da bisogni omogenei e simili comportamenti di acquisto. L'avvento dei Big Data è stato rivoluzionario in questo campo e ha modificato in maniera sostanziale l'approccio alla segmentazione di mercato. Se prima i dati di ogni consumatore erano difficilmente reperibili e comunque caratterizzati da un basso numero di attributi socio-demografici, con il paradigma dei Big Data si assiste a due fenomeni fondamentali: da un lato, è molto più semplice individuare i propri clienti grazie alla pervasività dei social network; dall'altro, ogni cliente è descritto da un elevato numero di attributi, come ad esempio i dati di profilo della propria pagina Facebook. Questa esplosione di informazioni rende più appetibile e potenzialmente più proficuo il processo di segmentazione, ma aggiunge, d'altro canto, un ulteriore livello di difficoltà. Gestire un grande insieme di clienti e individuare informazioni nascoste al suo interno diventa davvero difficoltoso senza l'ausilio di un procedimento automatico. È proprio a questo punto che la cluster analysis interviene a favore della segmentazione di mercato, fornendo una serie di strumenti che agevolano l'identificazione di caratteri emergenti dalla base di dati aziendale.
Essendo il modulo Right Time Personalization un prodotto di recente svi-

luppo, non è stato possibile disporre di un elevato numero di profili utente su cui condurre il lavoro. Il primo passo è stato, quindi, la creazione di una applicazione per la raccolta dei dati: abbiamo recuperato circa 25 mila profili Facebook, completi di numerose informazioni (anonimizzate) come l'età, il genere, l'istruzione, gli interessi personali (le pagine su cui si è indicato *like*) e relazioni di amicizia, creando un'unica rete sociale con attributi da cui sono partite le analisi e gli sviluppi principali del lavoro. Il passo successivo è stato quello di progettare e mettere a punto una infrastruttura per il preprocessing dei dati tramite procedure consolidate e di nuova generazione che tenessero conto delle peculiarità delle reti sociali: con questi strumenti il grafo iniziale è stato pulito, trasformato e infine campionato in un insieme di sottografi con caratteristiche topologiche e di profilo molto variegate: in questa maniera, abbiamo potuto disporre di una serie di dataset su cui valutare, nel modo più oggettivo possibile, le varie tecniche di clustering.
Dopo un approfondito studio della letteratura, abbiamo selezionato un insieme di algoritmi progettando un'interfaccia di trasformazione tra i sottografi a disposizione e il formato in input di ogni algoritmo: in questo modo è stato possibile condurre dei test pratici che misurassero le prestazioni delle varie tecniche di clustering, individuando quelle più promettenti e isolando quelle insoddisfacenti. Per tale scopo sono stati valutati molteplici indici di qualità ed i tempi d'esecuzione al variare dei parametri di ciascuna tecnica. In base a queste analisi abbiamo trovato metodi efficaci per il clustering di dataset medio-piccoli, altri computazionalmente più onerosi ma validi per il subspace clustering di grandi volumi di dati, nonché algoritmi su grafi che sfruttano e ottimizzano simultaneamente le relazioni e la somiglianza tra gli individui, gestendo nativamente i valori mancanti e la dimensionalità del problema. Infine, poiché ogni algoritmo richiede in ingresso almeno la specificazione del numero di cluster da individuare, abbiamo proposto delle soluzioni per la regolazione automatica dei parametri in input, delineando inoltre una serie di suggerimenti sugli scenari di successo e fallimento delle procedure a nostra disposizione.

5.2 Sviluppi futuri

La naturale continuazione di questo lavoro può partire dalla soluzione di varie problematiche legate ai dati. Come già affermato nel Capitolo 3, il modulo Right-Time Personalization è alimentata da tre sorgenti di dati: Facebook, Twitter e Foursquare. In questa tesi abbiamo lavorato con i profili Facebook perché l'informazione strutturata presente in essi ci ha permes-

so di focalizzare l'attenzione su aspetti più generali e specifici della cluster analysis. Integrare servizi come Twitter comporta, infatti, altrettante sfide: in questo contesto, il *tweet* è l'informazione principale da analizzare e il suo carattere non strutturato impone l'utilizzo di sofisticate tecniche di elaborazione del linguaggio naturale per estrarre la semantica ad essi associata ed individuare gli interessi dell'utente. Un ulteriore obiettivo da raggiungere sarebbe quello dell'*entity resolution*, ovvero l'identificazione e l'unificazione dei profili Facebook, Twitter e Foursquare che appartengono al medesimo individuo. Oltre alle difficoltà implementative di questa proposta ne incorrerebbero immediatamente altre teoriche, come l'esplosione della dimensionalità dei dati e la formulazione di un modello coerente che tenga conto delle diversità concettuali dei tre Social Network (si veda la sottosezione 3.1.1).

Un'altra prospettiva attraente riguarda l'automatizzazione della scelta dell'algoritmo: ci siamo chiesti, infatti, in che misura è possibile suggerire una tecnica di clustering su un certo dataset in ingresso, basandosi solamente sulle esecuzioni e le analisi fatte precedentemente, senza applicare tutti gli algoritmi a disposizione. Questo è un obiettivo tanto entusiasmante quanto delicato, infatti comporta una minuziosa attenzione a due questioni importanti: in primo luogo, come abbiamo sempre specificato nel corso del presente lavoro, un clustering è significativo solo se convalidato e, semmai, trasformato dal cliente; ciò comporta che se il framework suggerisce automaticamente l'utilizzo di un algoritmo con una certa parametrizzazione non vuol dire che il risultato sia da accettare in maniera "dogmatica". In secondo luogo, a complicare le cose, intervengono numerose sfide teoriche ed implementative. Se per il suggerimento di un algoritmo ci si vuole basare solamente sulle analisi fatte in precedenza, allora bisogna in qualche modo definire una misura di similarità tra insiemi di individui, per associare al dataset in esame il più simile dataset già processato e proporre per il primo le stesse considerazioni fatte sull'ultimo. Questo è un compito assai complicato sotto molteplici punti di vista. Sicuramente, per produrre risultati significativi è necessario aver memorizzato svariate analisi condotte su dataset altrettanto numerosi e il più possibile variegati dal punto di vista topologico e degli attributi di profilo. Conclusa questa fase resta comunque il problema della progettazione delle funzioni di similarità tra dataset: come abbiamo affermato nei capitoli precedenti, il nostro lavoro mette a disposizione un modulo per il calcolo di metriche topologiche che possono essere utilizzate per etichettare un grafo in ingresso e calcolare quanto esso si differenzi da quelli già processati. Però, per raggiungere a pieno l'obiettivo è necessario

stabilire anche in quale misura due dataset sono simili relativamente agli attributi e individuare, in definitiva, una funzione univoca che tenga conto contemporaneamente delle caratteristiche topologiche e di profilo dei dati in ingresso.

Sempre nell'ambito del marketing e delle dinamiche *social* potrebbe essere molto interessante arricchire il nostro lavoro con strumenti di *sentiment analysis* che svolgono oramai un ruolo di fondamentale importanza per recuperare automaticamente le opinioni dei clienti riguardo un prodotto o servizio. Utilizzando queste tecniche è possibile condurre una ulteriore segmentazione di mercato mirata ad intercettare il *mood* dei consumatori e intervenire a seconda della positività o negatività delle opinioni e del relativo grado di intensità emotiva. Questi aspetti possono avere ancor più valore del singolo *like* sulla pagina Facebook aziendale, e rappresentano per tanto una nuova frontiera per il monitoraggio della reputazione del brand e la pianificazione di innovative strategie di marketing.

In ultimo, citiamo una idea stimolante riguardo il *viral marketing* che può essere facilmente integrata nel nostro progetto. Supponiamo che una azienda stia cercando di promuovere un prodotto all'interno di una popolazione di potenziali clienti tramite l'offerta di campioni di prova gratuiti: è fondamentale chiedersi a chi proporre questa offerta. Gli studi sul *marketing virale* [21] affermano che c'è un vantaggio molto importante nel far conoscere un certo brand tramite il *passaparola* tra conoscenti: infatti un consumatore è più propenso all'acquisto se è stato consigliato da una persona fidata. Questo meccanismo, inoltre, innesca una cascata di segnalazioni che da un numero basso di utilizzatori iniziali può portare ad un ampio gruppo di clienti finali. Per questa ragione è essenziale individuare, all'interno di un grafo sociale, gli *opinion leader*, ovvero le persone più influenti dalle quali è possibile ottenere l'effetto cascata maggiore e proporre loro l'offerta dei campioni gratuiti [53]. Una prima soluzione può essere suggerita dalle metriche individuate dal modulo `Topological Metrics`, in particolare il grado e l'intermedietà di ciascun nodo. Inoltre, attraverso un algoritmo di clustering topologico ed overlapping potrebbero essere identificate le persone con alta connettività che appartengono a più comunità simultaneamente. Infine, una ulteriore sfida sarebbe quella di misurare la "reputazione" di ciascun individuo, che può essere approssimata con il suo grado nella rete sociale.

Bibliografia

[1] David A. Aaker. *Strategic Market Management.* Wiley, 10th edition, 2013.

[2] Micah Adler and Michael Mitzenmacher. Towards compressing web graphs. In *Proceedings of the Data Compression Conference*, DCC '01, pages 203–, Washington, DC, USA, 2001. IEEE Computer Society.

[3] Charu C. Aggarwal, Alexander Hinneburg, and Daniel A. Keim. On the surprising behavior of distance metrics in high dimensional space. In *Database Theory - ICDT 2001*, volume 1973, pages 420–434. Springer Berlin Heidelberg, 2001.

[4] Charu C. Aggarwal and Philip S. Yu. Finding generalized projected clusters in high dimensional spaces. In Weidong Chen, Jeffrey F. Naughton, and Philip A. Bernstein, editors, *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, May 16-18, 2000, Dallas, Texas, USA*, pages 70–81. ACM, 2000.

[5] Rakesh Agrawal, Johannes Gehrke, Dimitrios Gunopulos, and Prabhakar Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. *SIGMOD Rec.*, 27(2):94–105, June 1998.

[6] Edoardo M. Airoldi and Kathleen M. Carley. Sampling algorithms for pure network topologies. Technical report, 2005.

[7] Richard D. Alba. A graph-theoretic definition of a sociometric clique. *Journal of Mathematical Sociology*, 3:3–113, 1973.

[8] M. R. Anderberg. *Cluster Analysis for Applications.* Academic Press, 1973.

[9] Carol H. Anderson and Julian W. Vincze. *Strategic Marketing Management.* Houghton Mifflin Harcourt, 2nd edition, 2003.

[10] Yuichi Asahiro, Refael Hassin, and Kazuo Iwama. Complexity of finding dense subgraphs. *Discrete Appl. Math.*, 121(1-3):15–26, September 2002.

[11] Arindam Banerjee, Chase Krumpelman, Joydeep Ghosh, Sugato Basu, and Raymond J. Mooney. Model-based overlapping clustering. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, KDD '05, pages 532–537, New York, NY, USA, 2005. ACM.

[12] Albert-Laszlo Barabasi and Reka Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.

[13] Pavel Berkhin. A survey of clustering data mining techniques. *Grouping Multidimensional Data*, pages 25–71, 2006.

[14] Kevin Beyer, Jonathan Goldstein, Raghu Ramakrishnan, and Uri Shaft. When is nearest neighbor meaningful? In *In Int. Conf. on Database Theory*, pages 217–235, 1999.

[15] V.D. Blondel, J.L. Guillaume, R. Lambiotte, and E.L.J.S. Mech. Fast unfolding of communities in large networks. *J. Stat. Mech*, page P10008, 2008.

[16] Immanuel M. Bomze, Marco Budinich, Panos M. Pardalos, and Marcello Pelillo. The maximum clique problem. In *Handbook of Combinatorial Optimization*, pages 1–74. Kluwer Academic Publishers, 1999.

[17] Phillip Bonacich. Power and centrality: A family of measures. *American Journal of Sociology*, 92(5):1170–1182, 1987.

[18] Stephen Borgatti. The key player problem. In *Proceedings of CASOS 2002 Conference*, Pittsburgh, PA, 2002.

[19] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. OPTICS-OF: Identifying local outliers. In *Proceedings of the Third European Conference on Principles of Data Mining and Knowledge Discovery*, PKDD '99, pages 262–270, London, UK, UK, 1999. Springer-Verlag.

[20] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. LOF: Identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, SIGMOD '00, pages 93–104, New York, NY, USA, 2000. ACM.

[21] Jacqueline Johnson Brown and Peter H Reingen. Social Ties and Word-of-Mouth Referral Behavior. *Journal of Consumer Research*, 14(3):350–62, December 1987.

[22] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Outlier detection: A survey, 2007.

[23] Girish Chandrashekar and Ferat Sahin. A survey on feature selection methods. *Computers & Electrical Engineering*, 40(1):16 – 28, 2014. 40th-year commemorative issue.

[24] Manoranjan Dash and Huan Liu. Feature selection for clustering. pages 110–121, 2000.

[25] Thomas G. Dietterich. Ensemble methods in machine learning. In *Proceedings of the First International Workshop on Multiple Classifier Systems*, MCS '00, pages 1–15, London, UK, UK, 2000. Springer-Verlag.

[26] Carlotta Domeniconi, Dimitrios Gunopulos, Sheng Ma, Bojun Yan, Muna Al-Razgan, and Dimitris Papadopoulos. Locally adaptive metrics for clustering high dimensional data. *Data Min. Knowl. Discov.*, 14(1):63–97, February 2007.

[27] A. R. Donders, G. J. van der Heijden, T. Stijnen, and K. G. Moons. Review: a gentle introduction to imputation of missing values. *Journal of clinical epidemiology*, 59(10):1087–91, 2006.

[28] Levent Ertoz, Michael Steinbach, and Vipin Kumar. A new shared nearest neighbor clustering algorithm and its applications. In *Workshop on Clustering High Dimensional Data and its Applications at 2nd SIAM International Conference on Data Mining*, 2002.

[29] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. pages 226–231. AAAI Press, 1996.

[30] Santo Fortunato. Community detection in graphs. *Physics Reports*, 486:75–174, 2010.

[31] L.C. Freeman. Centrality in social networks conceptual clarification. *Social networks*, 1(3):215–239, 1979.

[32] Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.

[33] R. Ghaemi, N. Sulaiman, H. Ibrahim, and N. Mustapha. A Survey: Clustering Ensembles Techniques. *Proceedings of World Academy of Science, Engineering and Technology*, 38, February 2009.

[34] R. Gnanadesikan and J. R. Kettenring. Robust estimates, residuals, and outlier detection with multi-response data, 1972.

[35] Sanjay Goil, Harsha Nagesh, and Alok Choudhary. MAFIA: Efficient and scalable subspace clustering for very large data sets. Technical report, 1999.

[36] A.D. Gordon. *Classification*. Chapman and Hall, 2nd edition, 1999.

[37] Sander Greenland and William D. Finkle. A critical look at methods for handling missing covariates in epidemiologic regression analyses. *American Journal of Epidemiology*, 142(12):1255–1264, 1995.

[38] Frank E. Grubbs. Procedures for detecting outlying observations in samples. *Technometrics*, 11(1):1–21, February 1969.

[39] S. Guha, R. Rastogi, and Kyuseok Shim. ROCK: a robust clustering algorithm for categorical attributes. In *Data Engineering, 1999. Proceedings., 15th International Conference on*, pages 512–521, Mar 1999.

[40] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. CURE: An efficient clustering algorithm for large databases. *SIGMOD Rec.*, 27(2):73–84, June 1998.

[41] Stephan Günnemann, Brigitte Boden, and Thomas Seidl. DB-CSC: A density-based approach for subspace clustering in graphs with feature vectors. In Dimitrios Gunopulos, Thomas Hofmann, Donato Malerba, and Michalis Vazirgiannis, editors, *ECML/PKDD (1)*, volume 6911 of *Lecture Notes in Computer Science*, pages 565–580. Springer, 2011.

[42] Mark A. Hall. *Correlation-based Feature Selection for Machine Learning*. PhD thesis, 1999.

[43] Jiawei Han, Micheline Kamber, and Jian Pei. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edition, 2011.

[44] Mark S. Handcock, Adrian E. Raftery, and Jeremy M. Tantrum. Model-based clustering for social networks. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 170(2):301–354, 2007.

[45] Alexander Hinneburg and Daniel A. Keim. Optimal grid-clustering: Towards breaking the curse of dimensionality in high-dimensional clustering. In *Proceedings of the 25th International Conference on Very Large Data Bases*, VLDB '99, pages 506–517, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[46] Victoria Hodge and Jim Austin. A survey of outlier detection methodologies. *Artif. Intell. Rev.*, 22(2):85–126, October 2004.

[47] Woochang Hwang, Taehyong Kim, Murali Ramanathan, and Aidong Zhang. Bridging centrality: Graph mining from element level to group level. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, pages 336–344, New York, NY, USA, 2008. ACM.

[48] Paul Jaccard. Étude comparative de la distribution florale dans une portion des Alpes et des Jura. *Bulletin del la Société Vaudoise des Sciences Naturelles*, 37:547–579, 1901.

[49] A.K. Jain, R. P W Duin, and Jianchang Mao. Statistical pattern recognition: a review. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(1):4–37, Jan 2000.

[50] Anil K. Jain and Richard C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988.

[51] N. Karthikeyani Visalakshi and K. Thangavel. Impact of normalization in distributed k-means clustering. *International Journal of Soft Computing*, 4(4):168–172, 2009.

[52] George Karypis, Eui-Hong (Sam) Han, and Vipin Kumar. Chameleon: Hierarchical clustering using dynamic modeling. *Computer*, 32(8):68–75, August 1999.

[53] David Kempe, Jon Kleinberg, and Éva Tardos. Influential nodes in a diffusion model for social networks. pages 1127–1138, 2005.

[54] Edwin M. Knorr and Raymond T. Ng. A unified notion of outliers: Properties and computation. In *In Proc. of the International Conference on Knowledge Discovery and Data Mining*, pages 219–222. AAAI Press, 1997.

[55] Gueorgi Kossinets. Effects of missing data in social networks. *Social Networks*, 28:247–268, 2003.

[56] Hans-Peter Kriegel, Peer Kröger, and Arthur Zimek. Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. *ACM Trans. Knowl. Discov. Data*, 3(1):1:1–1:58, March 2009.

[57] V. Krishnamurthy, M. Faloutsos, M. Chrobak, L. Lao, J-H Cui, and A. G. Percus. Reducing large internet topologies for faster simulations. In *IN IFIP NETWORKING*, 2005.

[58] Douglas Laney. 3D data management: Controlling data volume, velocity and variety, February 2001.

[59] P. F. Lazarsfeld and R. K. Merton. Friendship as a social process: A substantive and methodological analysis. In M. Berger, T. Abel, and C. Page, editors, *Freedom and Control in Modern Society*, pages 18–66. Van Nostrand, New York, 1954.

[60] Jure Leskovec and Christos Faloutsos. Sampling from large graphs. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, pages 631–636, New York, NY, USA, 2006. ACM.

[61] Roderick J A Little and Donald B Rubin. *Statistical Analysis with Missing Data*. John Wiley & Sons, Inc., New York, NY, USA, 1986.

[62] Huan Liu and Lei Yu. Toward integrating feature selection algorithms for classification and clustering. *IEEE Trans. on Knowl. and Data Eng.*, 17(4):491–502, April 2005.

[63] R. Duncan Luce and Albert D. Perry. A method of matrix analysis of group structure. *Prychometrika*, October 1949.

[64] Oliver Mason and Mark Verwoerd. Graph theory and networks in biology. In *IET Systems Biology*, pages 89–119, 2007.

[65] Miller McPherson, Lynn Smith-Lovin, and James M Cook. Birds of a feather: Homophily in social networks. *Annual Review of Sociology*, 27(1):415–444, 2001.

[66] Nina Mishra, Robert Schreiber, Isabelle Stanton, and Robert E. Tarjan. Clustering social networks. In *Proceedings of the 5th International Conference on Algorithms and Models for the Web-graph*, WAW'07, pages 56–67, Berlin, Heidelberg, 2007. Springer-Verlag.

[67] Thomas M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1 edition, 1997.

[68] P. Mitra, C. A. Murthy, and S.K. Pal. Unsupervised feature selection using feature similarity. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(3):301–312, Mar 2002.

[69] Gabriela Moise, Arthur Zimek, Peer Kröger, Hans-Peter Kriegel, and Jörg Sander. Subspace and projected clustering: Experimental evaluation and analysis. *Knowl. Inf. Syst.*, 21(3):299–326, November 2009.

[70] L.C. Molina, L. Belanche, and A. Nebot. Feature selection algorithms: a survey and experimental evaluation. In *Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on*, pages 306–313, 2002.

[71] James H. Myers. *Segmentation & Positioning for Strategic Marketing Decisions*. South-Western Educational Pub, 1st edition, 1996.

[72] Shyam Varan Nath. Crime pattern detection using data mining. In Klaus-Dieter Althoff and Martin Schaaf, editors, *LWA*, volume 1/2006 of *Hildesheimer Informatik-Berichte*, pages 338–341. University of Hildesheim, Institute of Computer Science, 2006.

[73] M. E. J. Newman. The structure and function of complex networks. *SIAM REVIEW*, 45:167–256, 2003.

[74] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review*, E 69(026113), 2004.

[75] M. E. J. Newman and Juyong Park. Why social networks are different from other types of networks. *Phys. Rev. E*, 68:036122, September 2003.

[76] Mark E. Newman. Assortative mixing in networks. *Phys. Rev. Lett.*, 89(20):208701, 2002.

[77] Lance Parsons, Ehtesham Haque, and Huan Liu. Subspace clustering for high dimensional data: A review. *SIGKDD Explor. Newsl.*, 6(1):90–105, June 2004.

[78] Huiting Qiao, Mengqi Mei, Qi Zhang, Nian Liu, Xin Li, Qiaoyang Huang, and Deyu Li. The study of pet image segmentation using clustering algorithm. In Mian Long, editor, *World Congress on Medical*

Physics and Biomedical Engineering May 26-31, 2012, Beijing, China, volume 39 of *IFMBE Proceedings*, pages 1836–1839. Springer Berlin Heidelberg, 2013.

[79] Eréndira Rendón, Itzel Abundez, Alejandra Arizmendi, and Elvia M. Quiroz. Internal versus external cluster validation indexes. *International journal of computers and communication*, 5, 2011.

[80] Lior Rokach. Ensemble-based classifiers. *Artif. Intell. Rev.*, 33(1-2):1–39, February 2010.

[81] Peter Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *J. Comput. Appl. Math.*, 20(1):53–65, November 1987.

[82] D. B. Rubin. *Multiple Imputation for Nonresponse in Surveys*. Wiley, 1987.

[83] J.L. Schafer. *Analysis of Incomplete Multivariate Data*. Chapman and Hall, London, 1997.

[84] Eran Segal, Alexis Battle, and Daphne Koller. Decomposing gene expression into cellular processes. In *Pacific Symposium on Biocomputing*, pages 89–100, 2003.

[85] W. R. Smith. Product Differentiation and Market Segmentation as Alternative Marketing Strategies. 21(1):3–8+, 1956.

[86] Michael Steinbach, Levent Ertoz, and Vipin Kumar. Challenges of clustering high dimensional data. In L. T. Wille, editor, *New Vistas in Statistical Physics – Applications in Econophysics, Bioinformatics, and Pattern Recognition*. Springer-Verlag, 2003.

[87] Alexander Strehl and Joydeep Ghosh. Cluster ensembles — a knowledge reuse framework for combining multiple partitions. *J. Mach. Learn. Res.*, pages 583–617, March 2003.

[88] Yizhou Sun, Yintao Yu, and Jiawei Han. Ranking-based clustering of heterogeneous information networks with star network schema. In John F. Elder IV, Françoise Fogelman-Soulié, Peter A. Flach, and Mohammed Zaki, editors, *KDD*, pages 797–806. ACM, 2009.

[89] Richard J. Trudeau. *Introduction to Graph Theory*. Dover Publications, 1st edition, 1994.

[90] Werner Vach. *Logistic Regression with Missing Values in the Covariates*. Springer, 1994.

[91] René Vidal. A tutorial on subspace clustering.

[92] Stanley Wasserman and Katherine Faust. *Social network analysis: Methods and applications*, volume 8. Cambridge university press, 1994.

[93] D.J. Watts and S.H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, (393):440–442, 1998.

[94] H. White, S. Boorman, and R. Breiger. Social structure from multiple networks: I. blockmodels of roles and positions. *American Journal of Sociology*, 81(4):730–80, 1976.

[95] Zhiqiang Xu, Yiping Ke, Yi Wang, Hong Cheng, and James Cheng. A model-based approach to attributed graph clustering. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, SIGMOD '12, pages 505–516, New York, NY, USA, 2012. ACM.

[96] Jaewon Yang, Julian J. McAuley, and Jure Leskovec. Community detection in networks with node attributes. *CoRR*, abs/1401.7267, 2014.

[97] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. Birch: An efficient data clustering method for very large databases. *SIGMOD Rec.*, 25(2):103–114, June 1996.

[98] Zheng Zhao and Huan Liu. Spectral feature selection for supervised and unsupervised learning. In *Proceedings of the 24th International Conference on MachineLearning*, ICML '07, pages 1151–1157, New York, NY, USA, 2007. ACM.

[99] Yang Zhou, Hong Cheng, and Jeffrey Xu Yu. Clustering large attributed graphs: An efficient incremental approach. In Geoffrey I. Webb, Bing Liu 0001, Chengqi Zhang, Dimitrios Gunopulos, and Xindong Wu, editors, *ICDM*, pages 689–698. IEEE Computer Society, 2010.

Appendice A

Informazioni sui dataset

	utenti	archi *friend*	*like*	componenti
1	1258	70 967	112 707	1
2	1419	27 399	111 684	3
3	1390	18 469	106 785	2
4	1565	88 789	125 916	2
5	1469	48 023	139 691	2
6	1461	19 117	121 544	2
7	1579	52 330	124 374	3
8	1250	32 134	104 848	3
9	3091	115 695	222 839	4
10	3143	85 396	223 146	2
11	2633	56 785	195 395	4
12	2951	110 737	197 220	2
13	3166	61 634	204 507	2
14	2633	107 249	208 152	1
15	2997	90 389	191 949	1
16	3133	153 498	225 844	1
17	5690	147 368	340 145	1
18	5081	160 955	307 814	3
19	6398	426 429	353 303	3
20	5923	293 638	338 710	2
21	5966	288 784	340 986	1
22	5662	151 093	323 894	1
23	10880	472 228	533 653	2
24	11073	489 630	542 150	1
25	10730	413 476	550 188	2
26	10839	484 518	538 002	1

Tabella A.1: Proprietà dei campioni del dataset