Progetto Elaborazione Linguaggio Naturale: Tecniche di Clustering

Giuseppe De Palma

Alma Mater Studiorum - Università di Bologna
giuseppe.depalma@studio.unibo.it
Matricola: 854846

Sommario Ciaone

1 Introduzione

Il *clustering* (o analisi dei gruppi) è una forma di *machine learning* non supervisionato che permette di raggruppare in *cluster* elementi non annotati dati in input. Un cluster è una collezione di oggetti "simili" tra loro che sono "dissimili" rispetto agli oggetti degli altri cluster. Questo tipo di machine learning è ottimo per partizionare un insieme di dati in diverse "categorie", quindi poter eseguire diverse analisi ed ottenere nuove informazioni. Applicazioni tipiche in cui il clustering viene molto usato è il riconoscimento di email di spam (le email a scopi pubblicitari o di frode), oppure per l'aggregazione di notizie (Google News ne è un esempio).

Il clustering trova possibili applicazioni anche nel campo dell'elaborazione del linguaggio naturale. Oltre alle nuove possibili analisi sui corpora ed al fornire una visualizzazione pittografica delle parole raggrupate, un interessante utilizzo è quello della **generalizzazione** delle parole.

Possiamo considerare i vari cluster delle classi di equivalenza. Per questo motivo, se avessimo un dataset su cui comporre i cluster fatto di frasi e parole, allora si potrebbe assumere che una qualche parola che compare in una frase può essere sostituita con un'altra dello stesso cluster lasciando intatta la correttezza della frase. Ad esempio, se avessimo nel nostro dataset "per Lunedì", "per Martedì", "per Mercoledì", "per Sabato", "per Domenica", senza avere "per Giovedì" e "per Vernedì", e avessimo un cluster in cui i giorni della settimana sono raggruppati insieme, allora potremmo generalizzare l'utilizzo della preposizione "per" con Giovedì e Vernedì.

Il clustering, quindi, può essere molto utile anche nell'elaborazione del linguaggio naturale. Nel progetto in studio vengono testate le capacità di alcune tecniche di clustering da cui si derivano dei risultati per mostrarne le differenze, i pregi e i difetti. I dati utilizzati negli esperimenti, comunque, non sono parti di testo, ma semplici dataset di vettori numerici 2D in modo tale da poter facilmente visualizzare i grafici relativi ai cluster e determinare le caratteristiche di ogni tecnica.

1.1 Outline

[**SCRIVERE OUTLINE**]

2 Clustering

Ci sono numerosi algoritmi per effettuare clustering, ma essi sono classificabili in poche tipologie: il clustering gerarchico e il clustering partizionale. Clustering partizionale consiste nell'ottenere dei cluster, di solito in modo iterativo, ma spesso senza determinare una vera relazione tra gli elementi. Si inizia con un insieme di cluster iniziale ed iterativamente si riassegnano gli oggetti nei giusti cluster. Il clustering gerarchico, invece, forma un albero (la gerarchia) degli elementi dove un nodo rappresenta un sotto-cluster del nodo padre e le foglie sono i singoli oggetti.

Un'altra importante distinzione tra gli algoritmi di clustering è il *soft clustering* e *hard clustering*. Nel primo caso, ogni oggetto può essere assegnato a più cluster secondo un qualche grado di appartenenza, mentre nel secondo caso ogni oggetto è assegnato ad un unico cluster. In questo progetto vedremo quattro diversi algoritmi, due della classe di clustering gerarchico, due del clustering partizionale. I primi tre eseguono hard clustering mentre l'ultimo soft clustering.

Di seguito sono elencati i metodi implementati e testati:

- Clustering **gerarchico**
  1. Aggregativo
  2. Divisivo
- Clustering **partizionale**
  1. K-Means
  2. EM (soft clustering)

2.1 Gerarchico

Andando più in dettaglio sulle diverse tecniche, abbiamo detto che la prima classe di clustering permette di creare degli alberi con i cluster e sotto-cluster. Questo può essere ottenuto con un approcco *bottom-up* che è il clustering **aggregativo**, il quale inizia dai singoli oggetti e ne raggruppa i più simili, per poi raggruppare i gruppi più simili e così via, fino ad ottenere un unico gruppo che sarà la radice dell'albero. Un altro approccio è quello *top-down*, il clustering **divisivo**, che in modo inverso dal precedente parte dal gruppo comprendente tutti gli elementi e lo divide in sotto-gruppi in modo da massimizzare la similarità intrinseca dei gruppi, fino ad arrivare ai singoli elementi.

L'albero che si ottiene può essere visualizzato tramite un dendrogramma, come nella figura 1. Questo diagramma ci mostra le relazioni gerarchiche tra gli oggetti, infatti il suo principale utilizzo è quello di mostrare il modo migliore di organizzare i cluster.

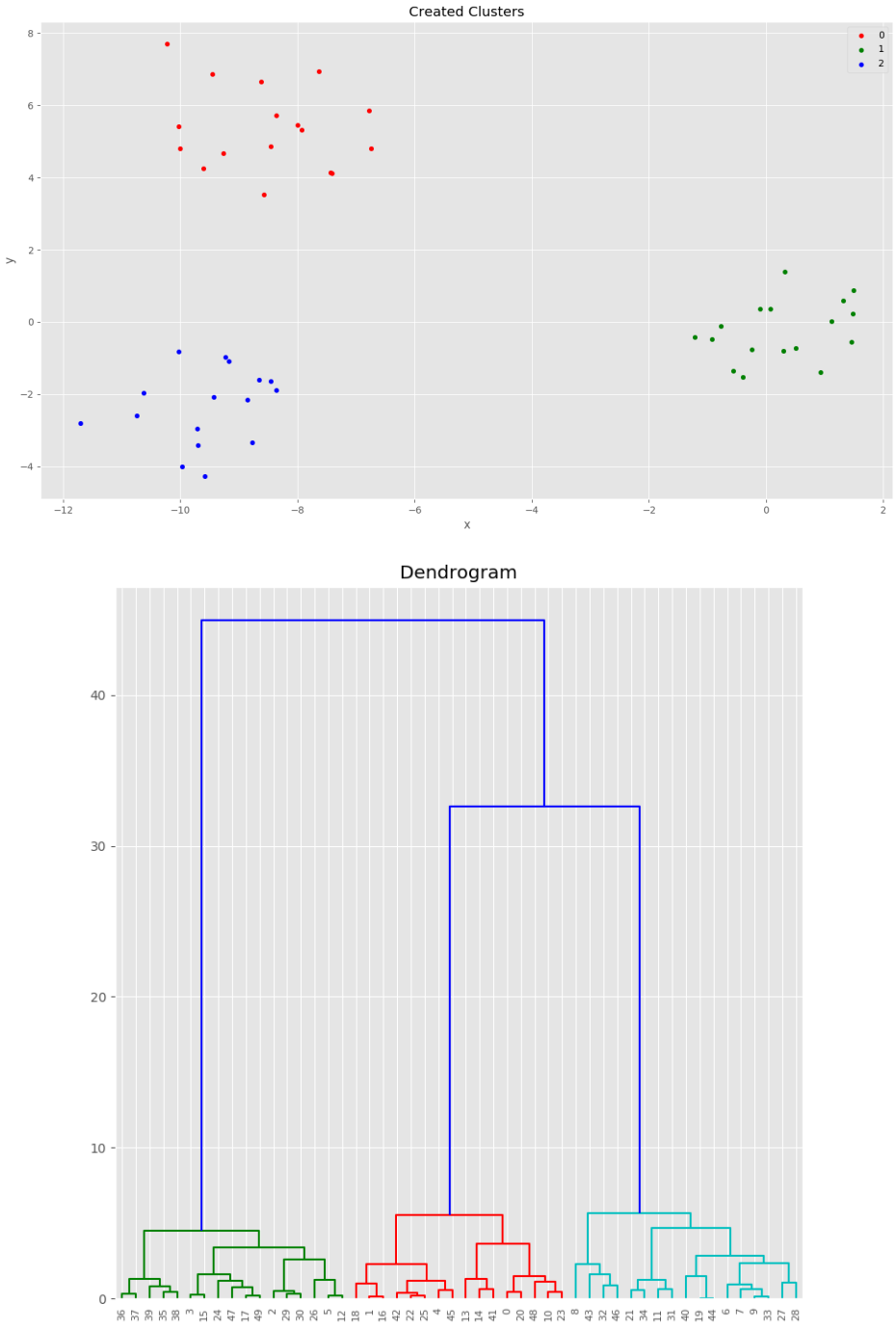

Figura 1: (a) insieme di 50 punti pre-divisi in 3 gruppi distinti, su cui si applica il clustering agglomerativo con la funzione di similarità "ward". Se ne ricava un dendogramma, in (b), che mostra le relazioni di quei dati. Si noti come tagliando orizzontalmente il cluster per le linee verticali più alte dove non appaiono linee orizzontali, rimangono proprio 3 sotto-cluster.

Concentrandosi sulle linee verticali, possiamo capire il numero ottimale di cluster da prefissare siccome l'asse y misura la somiglianza dei cluster, mentre l'asse x rappresenta gli oggetti e i cluster. Disegnando una linea orizzontale che "taglia" il dendrogramma, si afferma che i sotto-cluster al di sotto di questa linea saranno i raggruppamenti finali. Poichè linee verticali corte rappresentano gradi di similarità più alti tra gruppi, tagliando il dendrogramma per le linee verticali più alte che sono libere da linee orizzontali (quando c'è una unione di cluster), si ha il numero di cluster ottimale a cui assegnare gli elementi,

Aggregativo Questo tipo di clustering è realizzato tramite un algoritmo *greedy* che prende in input un insieme di dati S, da cui ogni oggetto è considerato essere un piccolo cluster da un elemento. Ad ogni passo l'algoritmo determina i due cluster più simili e li unisce in un nuovo cluster. L'algoritmo termina quando il cluster contenente tutti gli elementi di S viene formato, il quale sarà l'unico cluster rimanente. I modi in cui si possono determinare la similarità dei cluster sono molteplici. L'algoritmo fa uso di una funzione di similarità per calcolare quanto dei cluster sono simili tra di loro, ce ne sono diverse, ad esempio alcune molto utilizzate sono:

- single link: ottiene la similarità di due membri **più** simili da due cluster diversi, rispettivamente;
- complete link: ottiene la similarità dei due membri **meno** simili da due cluster diversi, rispettivamente;
- group-average: ottiene la similarità media tra i membri di due cluster.
- ward: **SPIEGARE**

Nonostante le funzioni di similarità possano differire anche ampiamente, una proprietà che tutte devono avere è la monotonia. Per un insieme di dati S e una funzione di similarità sim:

$$\forall c, c', \hat{c} \subseteq S.\ min(sim(c, c'), sim(c, \hat{c})) \geq sim(c, c' \cup \hat{c})$$

Questo perché l'operazione di unione garantisce di non aumentare la similarità, quindi una funzione che non obbedisce a questa condizione rovinerebbe la gerarchia in quanto cluster non simili, piazzati in parti lontane nell'albero, potrebbero ritrovarsi ad essere simili in unioni successive e perciò l'essere vicini nell'albero non corrisponderebbe più al concetto di similarità.

Divisivo Come per la controparte aggregativa, dietro il clustering divisivo c'è un algoritmo greedy. Invece di iniziare dai singoli elementi, si inizia dal cluster contenente tutti gli oggetti. Ad ogni iterazione si determina quale cluster è quello **meno** coerente e lo si divide in due. Come prima si utilizzano funzioni di similarità poichè due cluster con oggetti simili sono più coerenti di cluster con oggetti non simili. Ad esempio, un cluster con molti oggetti identici ha una coerenza massimale. L'operazione di divisione di un cluster è anch'essa una operazione di clustering, poichè bisogna trovare due sotto-cluster. Qualsiasi tecnica di clustering può essere usata per la divisione, anche il clustering aggregativo. Per queste

ragione il clustering divisivo è solitamente meno usato. However, there are tasks for which top-down clustering is the more nat-

2.2 Partizionale

Diversamente dal clustering appena discusso, gli algoritmi per quello partizionale spesso iniziano con una divisione casuale del dataset in vari cluster per poi raffinarli passo dopo passo attraverso la ricollocazione degli oggetti. Da questo procedimento varie domande sorgono.

1. Quando fermarsi con la ricollocazione?
2. In quanti cluster dividere il dataset?
3. In che modo si seleziona il miglior cluster in cui ricollocare un elemento?

Quando fermarsi può essere determinato tramite una metrica sulla qualità dei cluster. Questa può essere una funzione di similarità come la group-average e finchè il risualto incrementa di molto si continua la procedura.

La stessa metrica può essere utilizzata per rispondere alla seconda domanda. In alcuni casi possiamo già avere delle informazioni sul corretto numero di cluster da creare, oppure decidiamo arbitrariamente in quanti cluster dividere il dataset. In altri casi invece, possiamo dividere i dati in k cluster osservando l'andamento della metrica. Per capire un buon numero di cluster da avere bisogna vedere in quale transizione c'è un incremento sostanziale da $k-1$ a k cluster e un piccolo incremento da k a $k+1$ cluster. Trovando un numero k di cluster che abbia questa proprietà, possiamo avere in modo automatico una buona partizione del dataset.

Per l'ultima questione, sono gli algoritmi per il clustering partizionale che si occupano di come ricollocare gli oggetti, che ora verranno presentati.

K-Means K-Means è un algoritmo di hard clustering che definisce i cluster attraverso i loro punti centrali. Partendo da dei centroidi iniziali, iterativamente si assegna ogni oggetto al cluster il cui centroide è il più vicino, calcolato tramite la distanza euclidea. Una volta che tutti gli elementi sono stati assegnati, si ricalcolano i punti centrali di tutti i cluster e si ripete il procedimento di assegnazione così da raffinare il clustering.

EM L'algoritmo EM (Expectation-Maximization) è definibile come la versione di K-Means per il soft clustering. In quest'algoritmo possiamo vedere ogni cluster come una distribuzione di probabilità (in questo progetto vedremo l'algoritmo EM con *Gaussian mixture model*). L'idea è quindi di alternare due step: l'Expectation step dove accade il calcolo delle probabilità per ogni oggetto di essere stato generato da ognuna delle distribuzioni (in K-Means questo è analogo ad assegnare ogni punto ad un cluster). Successivamente c'è il Maximization step, in cui si massimizzano i parametri delle distribuzioni e degli oggetti (pesi, covarianza, medie) così da poter ripetere il primo step. L'equivalente in K-Means è muovere i punti centrali dei cluster per poi procedere con un'altra riallocazione.

3 Sessione Sperimentale

In questa sezione verranno mostrati vari esperimenti per sottolineare le differenze tra i quattro metodi presentati.

4 Conclusioni