Relazione progetto Data Mining:
Classificazione genetica lievito

Nicola Giuffrida - 28/06/2023

Introduzione

Lo scopo del paper studiato è quello di risolvere il problema della classificazione di dati le cui istanze possiedono o possono avere piu di una label, ovvero possono appartenere a piu di una classe.

Per risolvere questo problema ho applicato i metodi descritti nel paragrafo 4.1 del paper: **Problem Transformation** (PT) e **Binary Relevance**. Sono due approcci diversi al problema, il primo converte un dataset multi-label in uno single-label, trasformando cosi il problema in uno classificabile da un normale classificatore classico single-label; il metodo Binary Relevance invece converte un dataset con N classi in N dataset con una classe ciascuno.

Per calcolare l'accuratezza di ogni metodo ho usato la **jaccard similarity**:

che descrive la "similarità" tra una tupla del dataset originale e la tupla corrispondente

In the first step of a Jaccard Similarity measurement for two customers which consist of n binary attributes, the following four quantities (i.e., frequencies) are computed for the given binary data:

- a = the number of attributes that equal 1 for both objects i an j
- b = the number of attributes that equal 0 for object i but equal 1 for object j
- c = the number of attributes that equal 1 for object i but equal 0 for object j
- d = the number of attributes that equal 0 for both objects i and j.

Then, Jaccard Similarity for these attributes is calculated by the following equation:

$$J(i,j) = sim(i,j) = \frac{a}{a+b+c}$$

dopo essere stata trasformata da uno dei metodi citati e la/le cui classi sono state predette da un classificatore. Gli attributi utilizzari per il calcolo della jaccard similarity sono solo gli attributi classificatori.

Per tutti i PT ho utilizzato il classificatore **CART** mentre per il metodo Binary Relevance ho utilizzato il classificatore binario Support Vector Machines (**SVM**).

PT1

La soluzione PT1 proposta nel paper consiste nel **selezionare randomicamente una label per ogni istanza e rimuovere tutte le altre**. In questo modo si ottiene un dataset single-label ma ovviamente l'utilizzo di questo metodo comporta una importante perdita di informazioni. Questo perchè una volta che si è scelta la label di una istanza, **l'informazione riguardo tutte le altre label che tale istanza possedeva viene totalmente persa**.

Prima di procedere con la costruzione del modello e la classificazione qui come negli altri metodi ho convertito il dataset dal **multi-label format** (come indicato nel codice) al **single-label format**. Con questo non intendo la rimozione di classi, ovvero l'obiettivo di questo paper, ma soltanto del formato in cui esse sono rappresentate nal dataset; **invece di avere 14 attributi di classe ognuno con un valore che è 0 oppure 1, converto il dataset in uno equivalente in cui è presente un solo attributo di classe** ("Class" nel codice) il cui valore va da 1 a 14 e rappresenta la classe della istanza (soltanto una come definito in PT1). Questo perchè CART per poter operare sul dataset ha bisogno di uno ed un solo attributo classificatore, mentre il dataset dopo la conversione tramite PT1 è in multi-lable format ovvero possiede 14 attributi classificatori (le 14 classi originali) anche se in realtà solo una è presente in ogni istanza.

La jaccard similarity media ottenuta tramite PT1 è 0.18 circa.

PT2

PT2 prende come dataset quello ottenuto tramite PT1 ed **elimina ogni istanza simile ad un altra nel dataset**. Due istanze sono "simili" quando appartengono alla stessa classe. Il dataset ottenuto tramite PT2 possiede solo 14 istanze, ognuna di esse appartenente ad una diversa classe. Anche in questo caso si ha una grave perdita di informazioni (molto peggiore rispetto a quella di PT1) visto che **si perde il 99.2% delle istanze del dataset**, rendendo quindi molto inaccurato il classificatore.

La jaccard similarity media ottenuta tramite PT2 è 0.09 circa.

PT3

Il metodo PT3 risolve il problema della perdita di informazioni di PT1 e PT2. Le istanze del dataset ottenuto tramite PT3 **possiedono un attributo classificatore per ogni combinazione unica di classi presente nel dataset originale**. In questo modo, a differenza di PT1 e PT2 in cui ogni istanza appartiene ad una ed una sola classe originale, le istanze mantengono tutte le loro classi senza perdita di informazione. Viene però introdotto un nuovo problema: **il numero di attributi classificatori che si ottiene potrebbe essere molto vicino alla cardinalità dell'insieme delle parti di C**, dove C è l'insieme i cui elementi sono le classi originali del dataset. Questo significa che se il dataset originale possiede n classi e la loro distribuzione è piu o meno omogenea su tutto il dataset allora il numero di attributi classificatori ottenuto tende a 2^n, un numero ingestibile nel caso di molte classi. Il dataset che ho utilizzato nel progetto ha solo 14 classi ed esse sono distribuite in modo molto eterogeneo (ci sono poche classi molto frequenti e tutte le altre compaiono con una bassissima frequenza nel dataset) quindi non si avvicina a numeri molto alti, ma si ottengono comunque 200 attributi classificatori circa che sono abbastanza per far diminuire di molto l'accuratezza del classificatore e di consequenza la jaccard similarity. Inoltre un altro problema è che questo metodo non tiene conto della frequenza con cui le combinazioni di classi compaiono all'interno del dataset. Cio' significa che anche se una combinazione di classi compare soltanto una volta nelle 2400 istanze del dataset, essa verrà comunque aggiunta come attributo classificatore nel nuovo dataset, aumentando inutilmente il numero di attributi classificatori e compromettendo l'accuratezza del modello.

La jaccard similarity media ottenuta tramite PT3 è 0.05 circa. Questo valore cosi basso è appunto causato dal fatto che il dataset ottenuto ha un numero enorme di classi.

PT4

PT4 utilizza il dataset originale e non fa altro che aggiungere altre 14 classi, che rappreseno "l'opposto" di ogni classe. In particolare una istanza appartiene alla classe ¬Ln se e solo se tale istanza non apparteneva alla classe Ln. Tale trasformazione non aggiunge o rimuove alcuna informazione dal dataset. Il dataset ottenuto contiene le stesse informazioni (ma con 14 classi ridondanti aggiunte) e non risolve nemmeno il problema attenzionato dal paper. Infatti le istanze ottenute tramite PT4 sono ancora multi-label quindi totalmente inutilizzabili da qualunque classificatore single-label. Per questo motivo tale metodo non è implementabile.

Binary Relevance

Questo metodo, come dice il nome, **considera ogni classe singolarmente e sottopone ogni dataset ottenuto (contenente una delle 14 classi) ad un classificatore binario**, SVM in questo caso. In questo modo si risolve il problema di non considerare la frequenza con cui ogni classe compare nel dataset, non si ha una significativa perdita di informazione e il numero di classi totali rimane costante senza quindi impattare sull'accuratezza e consistenza del modello. Proprio per queste ragioni questo metodo è il migliore tra quelli osservati.

La jaccard similarity media ottenuta tramite Binary Relevance è 0.35 circa.

Conclusioni

Dai risultati ottenuti risulta evidente che il metodo Binary Relevance è migliore rispetto a tutti gli altri, esso risolve il problema di PT1 e PT2 di perdita di informazioni riguardo alle label eliminate e considera l'intero dataset e non un suo sottoinsieme per la classificazione. Inoltre non appesantisce il dataset aumentando il numero di attributi classificatori come in PT3 e PT4. L'accuratezza dei singoli classificatori binari di Binary Relevance è in media 80% che è un valore molto migliore rispetto a quelli ottenuti tramite CART nei vari PT. Ciò deriva dai motivi sopra citati e inoltre dal fatto che SVM offre una maggiore flessibilita perchè si puo scegliere, per ognuno dei 14 modelli, il kernel che offre la maggiore accuratezza quindi aumentando l'accuratezza totale del modello finale.