Naive Bayes Text Classifier

Caterina Giardi

January 2022

1 Introduzione

Il seguente progetto pone l'attenzione sul confronto di due algoritmi che effettuano una classificazione di testi. Gli algoritmi in questione sono il *Multi-Variate Bernoulli Naive Bayes Text Classifier*, e il *Multinomial Naive Bayes Text Classifier*, il cui funzionamento è descritto nel documento *A comparison of Event Models for Naive Bayes Text Classification (McCallum, Nigam)*. Il dataset usato contiene 20000 documenti suddivisi equamente in 20 categorie (https://www.cs.cmu.edu/ textlearning/).

2 Strumenti utilizzati

Nella realizzazione del progetto ho utilizzato:

- `nltk`, utilizzato per ottenere le stopwords, ossia parole non rilevanti al fine della classificazione
- `sklearn`, utilizzato per dividere il dataset in train e test
- `numpy`, per manipolare con facilità matrici e vettori
- `matplotlib`, per la riproduzione del grafico che confronta i due modelli
- `string`, per rimuovere la punteggiatura durante la tokenizzazione del dataset
- `pickle`, per salvare tutti i dati elaborati fino a quel momento e per ricaricarli al bisogno

3 Struttura del codice

Il progetto si divide in 4 file `.py`:

- `Multi-VariateBernoulliNB.py` che contiene la classe del classificatore omonimo.

- `MultinomialNB.py` che contiene la classe del classificatore omonimo.
- `aux_function.py` che contiene delle funzioni utili alla preparazione dei dati.
- `main.py` dove si allocano i classificatori e si chiama `predict()` per ottenere le accuratezze dei due classificatori.

Le due classi contengono al loro interno il metodo `prepare_data()`, il quale si occupa dell'analisi dei documenti e della feature selection; i metodi `calc_prior()`, `calc_p_of_w()`, `calc_p_of_d_given_c()` invece si occupano di calcolare le probabilità utili alla classificazione. Il metodo per selezionare le word del vocabolario utilizzando la feature selection è `get_features()`, e infine `predict()` che finalizza il calcolo delle probabilità e predice la classe di appartenenza dei documenti per poi chiamare `get_results()`, che stampa a video i risultati inerenti all'accuratezza del programma.

Oltre a questi, il progetto contiene anche dei file salvati a runtime al cui interno sono salvati valori di variabili, array, e matrici al fine di non doverli ricalcolare ogni volta che il programma viene eseguito.

4 Analisi dei documenti

Per procedere all'analisi dei documenti l'intera cartella `20_newsgroup` viene scorsa, i file vengono quindi memorizzati in una lista di liste: si avrà una lista di file per ogni categoria. Nella lista `filepath_list` sono contenuti tutti i path dei file, mentre nella lista `return_class` la classe di appartenenza di ogni documento. Si procede quindi alla divisione del dataset in training set e testing set, e poi alla tokenizzazione dei documenti, alla fine della quale otteniato un totale di circa 60'000 unique words. Tutto ciò avviene tramite alcune funzioni tra cui `clean_words()`, la quale pulisce le parole, `tokenizer()`, che assieme a `line_tokenizer()` trasforma ogni documento in una lista di parole.

Le probabilità per il calcolo delle features vengono eseguite una sola volta e danno frutto a due matrici (una per ogni modello) che verranno salvate coì da velocizzare le seguenti esecuzioni del programma.

5 Modello multivariato di Bernoulli e Modello multinomiale

Le classi create per creare i classificatori, differiscono solo per le formule utilizzate, infatti nel multinomiale ho usato

$$P(c_j|d_i) = log_{10}(Prior(c_j)) + \sum_{t=1}^{|V|} log_{10} P(w_t|c_j)^{N_{it}}$$

dove $P(w_t|c_j)$ è la probabilità che la parola t-esima compaia nella categoria j-esima, N_{it} è il numero di occorrenze di w_t del documento d_i;

mentre per Bernoulli ho usato

$$P(c_j|d_i) = log_{10}(Prior(c_j)) + \sum_{t=1}^{|V|} log_{10}[(B_{it}P(w_t|c_j) + (1 - B_{it})(1 - P(w_t|c_j))]$$

dove B_{it} vale 1 se w_t compare nel documento d_i e 0 altrimenti.

Le log probabilities garantiscono una maggior precisione e permettono la semplificazione dei calcoli, mantenendo valida l'uguaglianza.

6 Risultati

Nonostante la classificazione avvenga, risultati non sono molto soddisfacenti, l'accuratezza del modello Bernoulli non supera mai lo 0.7 mentre il modello multinomiale non supera mai lo 0.5, e non è evidente come l'accuratezza dei due aloritmi vari all'aumentare della taglia del vocabolario. Parte di ciò è causato dal dataset che non è completamente pulito: le unique words ottenute sono circa 15'000 in più rispetto a quelle riportate sul documento. Allego i risultati dell'accuratezza e il relativo grafico.

n features	multinomial accuracy	multi-variate bernoulli accuracy
20	0.213	0.299
50	0.228	0.388
100	0.275	0.471
500	0.333	0.565
750	0.348	0.580
1000	0.355	0.599
5000	0.416	0.618
10000	0.455	0.6175

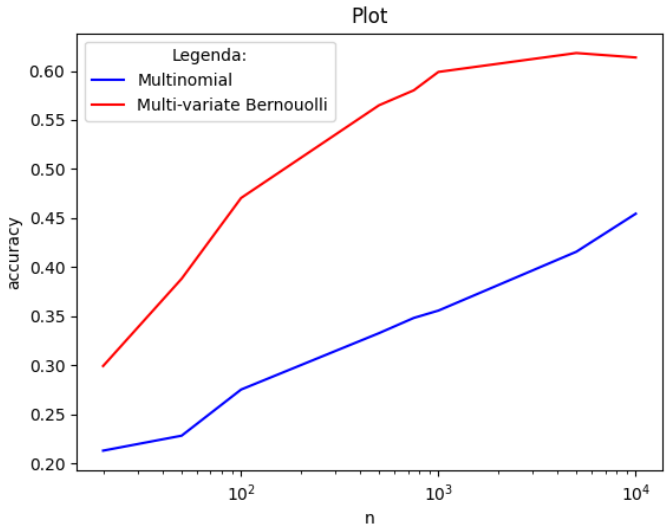

Figure 1: Caption