Relazione per l'esame di Intelligenza Artificiale

Giulia Bertazzini 6320125

16 Aprile 2020

1 Introduzione

Nel machine learning, il Perceptron è un algoritmo di apprendimento supervisionato per la classificazione binaria. È un tipo di classificatore lineare, ovvero un algoritmo di classificazione che fa le sue previsioni basate su una funzione di predizione lineare, la quale combina un insieme di pesi con il vettore di caratteristiche. Un modo più intuitivo di pensare è come una rete neurale con un solo neurone che accetta uno o più input, li processa e restituisce un output.
La convergenza del Perceptron, e dunque la sua capacità di predire correttamente la tipologia di un campione, sono strettamente legati alla struttura del dataset stesso. È necessario infatti che le classi siano linearmente separabili e, se ciò non fosse possibile, l'algoritmo non potrà mai giungere ad una stabilità, continuando di fatto ad aggiornare i suoi pesi senza soluzione di continuità.

In questo elaborato si è utilizzato l'algoritmo per classificare documenti testuali, utilizzando concretamente due datasets distinti: 20 newsgroups e Reuters-21578

2 Librerie

Il progetto è stato interamente implementato nel linguaggio Python. È stata utilizzata la libreria scikit-learn nei file newsgroups.py e reuters.py per:

- Caricare i datasets
- Utilizzare l'implementazione dell'algoritmo Perceptron
- Estrarre la bag of words
- Calcolare l'accuratezza

È stata poi usata la libreria matplotlib per rappresentare graficamente l'accuratezza della classificazione al variare della dimensione del vocabolario e del numero di esempi.

3 I datasets

Il dataset 20 newsgroups si compone di circa 20mila documenti testuali, dove ognuno fa parte di una certa categoria tra 20 possibili. Il caricamento del dataset è stato effettuato attraverso una funzione resa disponibile dalla libreria sklearn, *sklearn.dataset.fetch_20newsgroups()*.
Tale funzione presenta tra i parametri "subset", che permette di specificare quale tipo di dataset vogliamo caricare (train o test), "categories" per elencare particolari categorie del dataset a cui vogliamo fare riferimento (se non specificato, vengono caricate tutte le categorie) e "remove" che permette di rimuovere headers, footers e quotes.

Il dataset Reuters-21578 è costituito invece da 21 file in formato SGM contenenti ciascuno 1000 documenti, ad eccezione dell'ultimo che ne contiene 578. Come richiesto dal problema, per questo dataset sono stati utilizzati soltanto documenti appartenenti alle 10 categorie più frequenti, attraverso un processo di filtraggio e di analisi dei tag, presenti nel file preprocessing.py.

Ogni documento in ciascun file.SGM è strutturato in una gerarchia di tag annidati, per la cui descrizione più dettagliata si rimanda alla documentazione di Reuters-21578.

3.1 Preprocessing Reuters-21578

Per estrarre la lista delle 10 categorie più frequenti, la funzione top10categories() in preprocessing.py analizza le occorrenze di ciascuna categoria per ogni file, ricavando da all-topics-string.lc.txt la lista di tutte e 135 le categorie presenti nei documenti e successivamente contando in ogni file.SGM l'occorrenza del tag del tipo <D>nome-categoria</D> per ogni nome-categoria.

Successivamente è stato organizzato il dataset Reuters-21578 in più cartelle per ognuna delle 10 categorie più frequenti, analizzando ancora una volta i tags presenti di ciascun file: in questo caso il tag <TOPICS> </TOPICS> è servito per verificare l'appartenenza a una categoria, mentre it tag </REUTERS> è stato utilizzato per dividere i documenti.

In questo processo è stato simulato il "Modified Apte Split" descritto nel file READMEReuters.txt non considerando documenti che non ne facessero parte al fine di allineare i risultati di questo esperimento su Reuters-21578 a tutti quelli che abbiano adottato tale Split.

The Modified Apte ("ModApte") Split:
Training Set: LEWISSPLIT="TRAIN"; TOPICS="YES"
Test Set: LEWISSPLIT="TEST"; TOPICS="YES"
Unused: LEWISSPLIT="NOT-USED"; TOPICS="YES"
or TOPICS="NO"
or TOPICS=_BYPASS"

Nel caso di testi in cui sono presenti più tag di categorie facenti parte della lista restituita da top10categories(), viene duplicato l' esempio tante volte quante sono le categorie in questione e viene inserito un duplicato in ognuna delle classi di appartenenza. Si verifica perciò un aumento degli esempi.

Una volta trovate le 10 categorie più frequenti, si è utilizzato la funzione di scikit-learn *sklearn.dataset.load files()* per caricare il dataset.

4 Bag of words

Per l'estrarre la bag of words di ciascun dataset è stata utilizzata la funzione CountVectorizer() che restituisce una rappresentazione matriciale dell'intero dataset, costruendo un dizionario di tutte le parole presenti nel dataset per ogni documento e salvando le feature di ciascuna parola. Tramite il parametro max_features è possibile considerare soltanto un numero limitato di parole del vocabolario (quelle più frequenti) e inoltre sono state eliminate le parole presenti nella lista stop-words='english'.

5 Risultati e conclusioni

In questo paragrafo sono riportati i risultati ottenuti al variare della dimensione del vocabolario e del numero di esempi.
Si osserva che l'algoritmo funziona meglio per entrambi i datasets all'aumentare della dimensione del vocabolario. Tuttavia, mentre nel caso del 20 newsgroups con un vocabulary size di 100 parole si riesce a ottenere un'accuratezza di quasi il 60%, per il secondo dataset sono necessarie più di 1000 parole per ottenere circa tale accuratezza.
Per quanto riguarda il numero di esempi invece all'aumentare di questi si nota un peggioramento in termini di accuratezza per entrambi i datasets, peggioramento che comunque è sempre più elevato per il dataset Reuters.

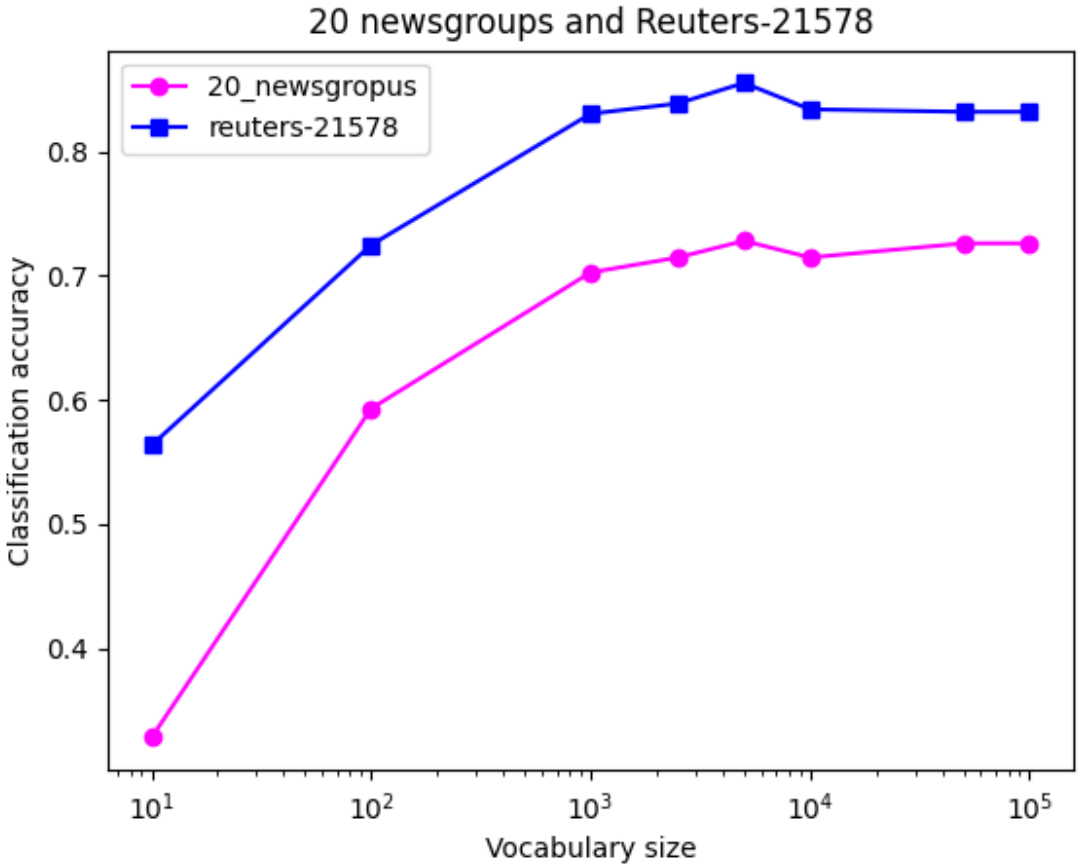

Figura 1: rappresentazione dell'accuratezza al variare della dimensione del vocabolario in entrambi i datasets

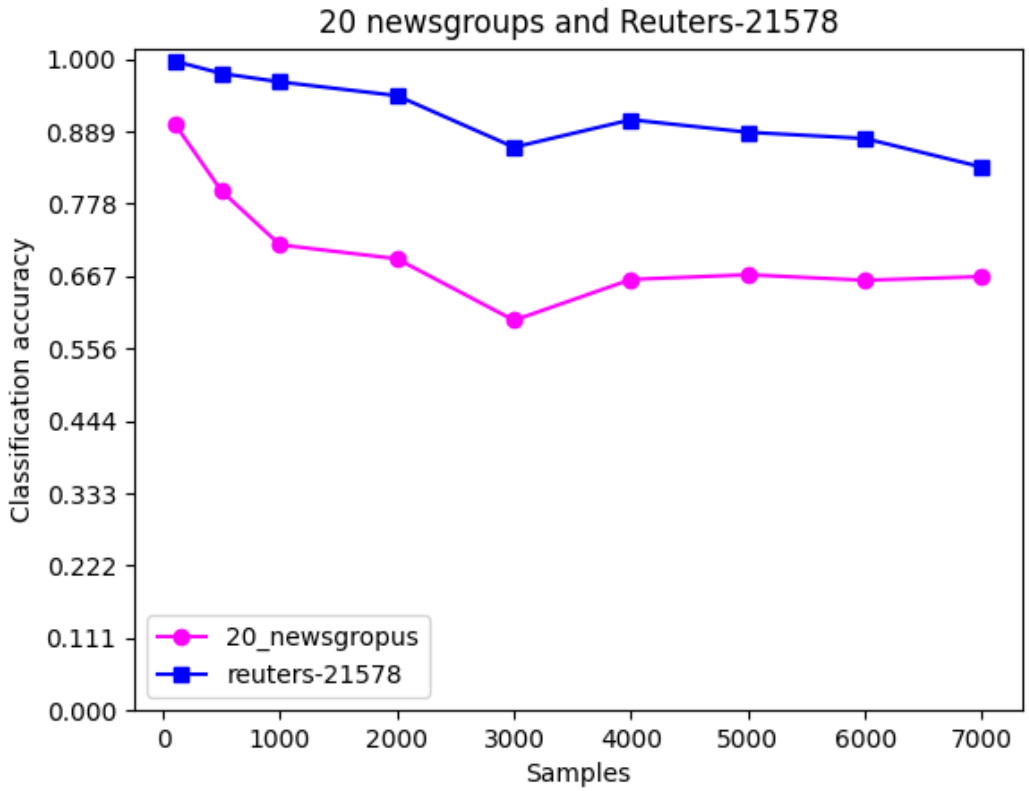

Figura 2: rappresentazione dell'accuratezza all'aumentare
del numero di esempi in entrambi i datasets

6 References

[1] Stuart Russell and Peter Norvig. Artificial Intelligence: A Modern Approach.
3rd edition. Pearson, 2010

[2] A. McCallum and K. Nigam (1998). A comparison of event models for Naive Bayes text classification

[3] Gavin Hackeling. Mastering Machine Learning with scikit-learn. Packt Publishing Ltd, 2014

[4] scikit-learn documentation: http://scikit-learn.org/stable/user
guide.html