Algoritmo Naive Bayes per la classificazione delle email

Manuel Salamino

Aprile 2018

Lo scopo del programma implementato é quello di, dato un dataset contenente delle email divise per categoria in base all'argomento trattato, riuscire a progettare un **classificatore di documenti**, e quindi fare in modo che dopo la fase di training, in cui il programma viene istruito sui parametri necessari a compiere questo riconoscimento, il codice fosse in grado di, date delle nuove email, riuscire a classificarle correttamente.
In questo programma era stato stato richiesto di usare l'Algoritmo Naive Bayes nella versione Bernoulli, quindi l'unico risultato che puó essere riportato é se l'email testata risultati positiva o negativa. Quindi é scelta una categoria come positiva e le rimanenti come negative, in questo modo se il documento risultata positivo posso dire che appartiene alla classe positiva, altrimenti appartiene ad una delle classi negative.

Dopodiché era richiesto di calcolare la Curva ROC, utilizzando i parametri richiesti, in modo da osservare l'efficacia del programma sviluppato.

1 Documentazione del codice

1.1 Classifier.py

Contiene l'implementazione di tutto l'algoritmo Naive Bayes, sia del train che del test.
É composto da piú funzioni, ognuna delle quali si occupa di svolgere uno dei passaggi neccessari a fare la classificazione.
Al suo interno ci sono anche delle funzioni (*createData* e *hyperplane*) necessarie a calcolare l'iperpiano di separazione della classe negativa da quella positiva, necessario a sua volta per calcolare i parametri da passare allo script che si occupa di calcolare la Curva ROC.
Per il calcolo dei parametri relativi all'iperpiano di seperazione sono state usate funzioni fornite dalla libreria *scikit learn*

1.2 ROCCurve.py

Composto da due sole funzioni: *calcolateActual(path)* e *rocCurve(...)*.

- *calcolateActual(path)* si occupa di, una volta ricevuto il percorso della classe scelta come positiva, fare una lista in cui compare **0** in corrispondenza delle email appartenenti a classi negative e **1** se appartengono alla classe positiva.
  Questa funzione quindi restituisce una lista composta da soli 0 e 1.
- *rocCurve(...)* si occupa di calcolare la curva ROC attraverso delle apposite funzioni della libreria *scikit learn*, che prendono in ingresso due liste e restituiscono il grafico che rappresentea la curva ROC.

Le liste prese in ingresso da quest'ultima funzione sono:

- la lista restituita da *calcolateActual(path)*.
- una lista contenente, per ogni email da testare, la distanza dall'iperpiano di separazione, questo secondo paramentro rappresenta lo *score* per calcolare la curva ROC.

Importante: per tracciare in modo corretto la curva ROC é necessario che l'**ordine della cartelle** in cui sono le email da testare, e l'**ordine delle email** stesse all'interno della cartella rimanga **invariato**. Perché il calcolo viene fatto assumendo che l'*i-esimo* elemento della prima e della seconda lista siano relativi all'*i-esimo* documento, prendendo in ordine le cartelle e le email al suo interno.
Dunque se non venisse rispettata questa condizione la curva ROC disegnata perderebbe significato.

1.3 Dataset.py

Quí sono contenute le classi necessarie a leggere le email presenti nel dataset, per modellarlo e rendere compatibili le funzioni appena descritte con i dati da analizzare.
Nello specifico ci sono due classi:

- *Mail*, serve a leggere le email e creare una lista che contiene tutti le parole che soddisfano determinati requsiti (come avere una certa lunghezze, per evitare di prendere in considerazione congiunzioni, proposizioni ecc e non devono appartenere alla lista *commmonWords* dove ho inserito alcune delle parole che apparivano piú frequentemente nelle email e che non avevano alcune rilevanza in termini di classificazioni, per esempio avverbi ecc), in questo attributo *text* ci sono le parole contenute nel testo effettivo dell'email (escluso quindi l'header) e la parole che compongono l'Oggetto (*Subject*) dell'email perché al suo interno ci sono delle parole chiave la cui presenza risulta fondamentale per riconoscere la categoria a cui una email appartiene.
- *Class*, contiene i dati relativi ad una categoria/argomento di email ed al suo interno é presente una lista di oggetti *Mail* che hanno come contenuto appunto le parole delle email presenti in quella determinata categoria.

1.4 Exec.py

É lo script da eseguire per lanciare l'esecuzione di tutto il programma.
Non fa altro che definire il percorso della cartella (che corrisponde ad una classe) da prendere come positiva, prima nel train e poi nel test (naturalmente in entrambi i casi la classe da definire come positiva dovrá essere la stessa) e poi lancia la funzione *classifier()* che restituirá la lista contenente i paramentri da passare come score nel calcolo della curva ROC, e poi lancia appunto la funzione *curveROC()* che calcola e traccia il grafico della curva ROC di quello specifico caso.

Questo procedimento, come richiesto nel testo dell'esercizio, viene ripetuto per tre volte prendendo ogni volta una classe positiva differente per confrontare i risultati delle tre prove.

2 Descrizione Dataset utilizzato

Per l'esecuzione del programma il Dataset utilizzato é **20 newsgroups**, che é giá suddiviso in train and test, ed ognuno di questi é suddiviso a sua volta in 20 cartelle ognuna delle quali contiene email su un certo argomento.
Per permettere il corretto funzionamento del test, in queste due suddivisioni ci saranno le stesse categori ed il numero di email presenti nelle cartelle del train sará maggiore del numero di email presenti nelle cartelle del test.

3 Come usare il codice

Come detto precedentemente il file da eseguire é *Exec.py* ed al suo interno sono importate le funzioni presenti sugli altri file e quindi semplicemente con una chiamata di funzione si riesce ad eseguire la classificazione di documenti ed a tracciare il grafico della curva ROC.
Entrambe le funzioni (*classifier()* e *curveROC()*) vogliono come parametro un percorso (*path*) che indica la classe/cartella che contiene le email della categoria scelta come positiva.
Nelle esecuzioni svolte da me il dataset é stato posizionato nella stessa cartella dove risiedono i file *.py* all'interno della cartella "*dataset*" e quindi riuscivo ad accedervi attraverso il percorso *'dataset/20news-bydate-train/nomeCartella'* per quanto riguarda il train, e *'dataset/20news-bydate-test/nomeCartella'* per quanto riguarda il test.

Sono necessari entrambi questi percorsi poiché alla funzione *classifier()* si deve passare il percorso della cartella su cui fare il train, mentre alla funzione *roccurve()* il percorso da passare sará relativo alla cartella su cui fare il test in modo da ricavarsi dinamicamente i valore reali della classificazione, poi a questa seconda funzione andrá anche passata la lista contenente gli altri paramentri per disegnare la curca ROC, che sono quelli restituiti da *classifier()*.

4 Risultati Ottenuti

L'esempio di esecuzione svolto é stato fatto prendendo come primo test la cartella *"alt.atheism"* e quindi passando come percorso del train *'dataset/20news-bydate-train/ alt.atheism'* e come percorso del test *'dataset/20news-bydate-test/alt.atheism'*.
Il **tempo di esecuzione** totale di tutte le operazioni (train, test, calcolo iperpiano, calcolo distanza dall'iperpiano e calcolo del grafico) per questo **primo test** é di **17 minuti circa**.

Il programma come risultato, dopo aver eseguito train e test, restituirá anche un resoconto del risultato del test, che raccoglie i dati relativi ad ogni cartella dicendo quante email risultato positive e quante negative in essa. Il risultato che ci si aspetta é che nella cartella scelta come positiva la maggior parte delle email risultino positive, viceversa nelle restanti cartelle:

Tabella 1: Risultati del primo test effettuato.

Categoria	**N email positive**	**N email negative**
alt.atheism	224	95
comp.graphics	58	331
comp.os.ms-windows.misc	32	362
comp.sys.ibm.pc.hardware	26	366
comp.sys.mac.hardware	30	355
comp.windows.x	42	353
misc.forsale	111	279
rec.autos	54	342
rec.motorcycles	62	336
rec.sport.baseball	76	321
rec.sport.hockey	64	335
sci.crypt	32	364
sci.electronics	54	339
sci.med	107	289
sci.space	64	330
soc.religion.christian	151	247
talk.politics.guns	53	311
talk.politics.mideast	100	276
talk.politics.misc	56	254
talk.religion.misc	94	157

Dopo la tabella con questi dati viene calcolata la curva ROC come descritto precedentemente, ed il grafico risultante é il seguente:

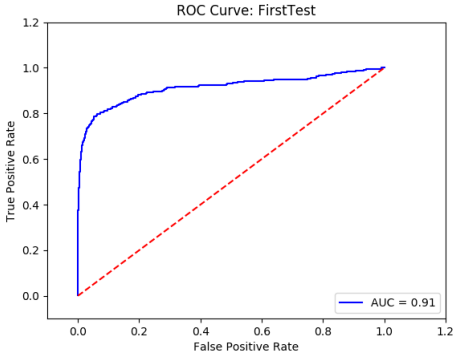

Figure 1: *Curva ROC del primo test effettuato.*

É necessario precisare che ai fini di poter eseguire l'algoritmo di classificazione in tempi accettabili durante la fase di train, dopo aver letto tutte le email ed aver contato ogni parola in quanti documenti appare, é stata eseguita una riduzione della taglia del vocabolario (l'insieme di tutte le parole trovare nei documenti).
Nei documenti positivi le parole trovate sono state tenute tutte, ad eccezione di quelle che sono state scartate in fase di lettura del documento, mentre per i documenti negativi sono state scartate le parole che non raggiungevano una certa soglia, rappresentata dal numero di documenti in cui queste parole comparivano.
Date che, considerando il nostro dataset, le email positive sottoposte alla fase di train erano circa 500, mentre quelle negative circa 10000 il criterio scelto per fare questa riduzione é stato quello di tenere soltanto la parole che appaiono in almeno lo 0,66% dei documenti, risultato ottenuto dopo varie prove. Poiché se la soglia veniva alzata (quindi rimosse piú parole) aumentavano i casi di email positive, viceversa se la soglia veniva abbassata, dunque dopo varie prove queste é risultata essere la soglia piú idonea.

Successivamente vengono fatte anche altre due stampe che prenderanno in esame altre due cartelle, rispettivamente *"rec.motorcycles"* e *"sci.electronics"*, dove i tempi di esecuzione ed i risultati sono pressoché uguali. Naturalmente cambierá la cartella in cui il numero di email positive risulta in maggioranza e ci sará un variazione nei singoli valori, ma rimanendo all'incirca nello stesso range.

5 Conclusioni

Alla luce dei risultati ottenuti effettuando questi test possiamo notare che la maggioranza degli esiti all'interno delle classi é netta, peró si ha comunque un numero di errori abbastanza alto.
Nei risultati dell'esempio sopra riportato possiamo osservare che avendo scelto come positiva la classe *atheism* (quindi ha a che fare con l'ateismo) si ha un numero di campioni positivi molto alto anche per la classe *christian* (che quindi avrá a che fare con il cristianesimo) dato che entrambi trattano argomenti affini.
Mentre quando si parla ad esempio di *hardware* i numeri si abbassano notevolmente.

Per quanto rigurda la curva ROC risutante dimostra che i risultati nel complesso sono buoni, perché piú la curva si avvicina all'angolo in alto a sinistra migliori sono le prestazioni, perché significa che é alto il tasso di vero positivo e basso quello di falso positivo.