

# Classificazione di testo con Perceptron

Carlo Baronti

Università degli Studi di Firenze

Email: carlo.baronti@stud.unifi.it

**Sommario**—Il lavoro consiste nell’implementazione e analisi delle performance del classificatore di testo tramite l’algoritmo Perceptron. Come dataset di riferimento è stato usato il ModApte split di Reuters-21578 di cui sono state scelte 6 categorie.

## 1. Introduzione

Inizialmente sono stati divisi i documenti di training e i documenti di test, dopo di che è stato estrapolato un insieme di parole detto *vocabolario*, da cui sono state eliminate le stopwords, per fare ciò ci si è avvalsi dello stemmer di Porter. Tali funzioni preliminari vengono eseguite in circa un minuto su Google Colab che continueremo ad utilizzare come riferimento per i tempi di esecuzione.

I risultati presentati in questo lavoro si riferiscono solo alla categoria *earn*, mentre i risultati relativi alle altre categorie sono accessibili presso [4]

## 2. ModApte split Reuters-21578 dataset

Delle 90 categorie presenti nel dataset Reuters-21578 ne sono state selezionate 6. Si riportano il numero di documenti di training e test per ognuna delle categorie selezionate:

- *acq*: train 1650, test 719
- *earn*: train 2877, test 1087
- *money-fx*: train 538, test 179
- *grain*: train 433, test 149
- *crude*: train 389, test 189
- *trade*: train 368, test 117

## 3. Definizioni

Nel testo si farà uso delle seguenti definizioni:

- $d$ : documento
- $|D|$ : numero di documenti
- $C$ : categoria
- $V$ : vocabolario
- $V_c$ : vocabolario della categoria
- $w$ : parola
- $N_{cw}$ : numero di documenti nella categoria  $c$  che contengono la parola  $w$
- $N_c$ : numero documenti nella categoria  $c$

## 4. Perceptron

Perceptron rappresenta un concetto di neurone. Fa uso del concetto di regressione lineare. Occorre prima però dare un’adeguata rappresentazione ai documenti.

### 4.1. Rappresentazione documenti

Dopo aver proceduto all’estrazione del vocabolario, alla suddivisione dei documenti di training e test, e all’ulteriore suddivisione di tali documenti per categorie, per ogni documento viene estratta una sua rappresentazione TF-IDF (*Term Frequency-Inverse Document Frequency*). Tale rappresentazione consiste nel pesare la rilevanza di un dato documento nei confronti di una certa parola prendendo in considerazione due fattori:

- 1) la frequenza di una parola nel documento (TF)
- 2) l’importanza generale di una parola nell’insieme dei documenti considerati (IDF)

Moltiplicando la IDF relativa a una parola  $w$  con la corrispondente TF, nel documento  $d$ , si ottiene un vettore di lunghezza pari al numero di parole presenti nel vocabolario, in cui ogni elemento del vettore è il prodotto  $tf[w] * idf[w]$ .

**4.1.1. Frequency.** Nello specifico la frequenza della  $i$ -esima parola nel  $j$ -esimo documento  $j$  risulta

$$tf_{i,j} = \frac{n_{i,j}}{|d_j|} \quad (1)$$

dove  $n_{i,j}$  rappresenta il numero di occorrenze della parola  $i$ -esima nel  $j$ -esimo documento.

**4.1.2. Document Frequency.** La IDF, come detto, misura l’importanza di una parola nell’insieme dei documenti considerati:

$$idf_i = \log \frac{|D|}{|\{d : w_i \in d\}|} \quad (2)$$

dove  $|D|$  rappresenta il numero di documenti di training e il denominatore rappresenta il numero di documenti che contengono la  $i$ -esima parola  $w_i$  del vocabolario  $V$ .

**4.1.3. TF-IDF.** Di conseguenza la rappresentazione TF-IDF per la parola  $i$ -esima  $w_i$  e il documento  $j$ -esimo risulta

$$tfidf_{i,j} = tf_{i,j} \cdot idf_i \quad (3)$$

## 4.2. Algoritmo

L'algoritmo suddivide lo spazio dei documenti in due parti, positiva e negativa, tramite un iperpiano. Tale iperpiano è formato dai vettori dei pesi (*weights*) e dei *bias*. L'aggiornamento viene effettuato in base al risultato di

$$y_i(w \odot x) \quad (4)$$

dove

- 1)  $y_i$  ha valore in  $\{-1, 1\}$  a seconda che il documento  $i$ -esimo appartenga o meno alla categoria esaminata
- 2)  $w$  è il vettore dei pesi corrente
- 3)  $x$  è la rappresentazione TF-IDF del documento considerato

Pesi e bias vengono quindi aggiornati se la classificazione commette un errore, cioè quando l'equazione (4) ha valore minore di zero. Per i dettagli delle regole di aggiornamento si fa riferimento a [1].

## 4.3. Precision and Recall

Precision e Recall sono valori che riguardano l'accuratezza dell'algoritmo e sono così definite:

Precision	$\frac{TP}{TP+FP}$
Recall	$\frac{TP}{TP+FN}$

- TP = Veri Positivi
- FP = Falsi Positivi
- FN = Falsi Negativi

## 4.4. Risultati e tempi di esecuzione

Nell'osservare i tempi di esecuzione del Perceptron c'è da tenere conto che

$$t_{perceptron} = t_{init} + t_{tf} + t_{idf} + t_{tfidf} + t_{train} \quad (5)$$

dove

- $t_{init}$  : tempo di inizializzazione
- $t_{tf}$  : tempo per il calcolo della TF
- $t_{idf}$  : tempo per il calcolo della IDF
- $t_{train}$  : tempo per training

Il training richiede un tempo variabile a seconda del numero massimo di iterazioni impostogli, tuttavia effettuando 10000 iterazioni,  $t_{init} \approx 2$  min,  $t_{tf} \approx 30$  sec,  $t_{idf} \approx 3$  min,  $t_{tfidf} \approx 3$  min,  $t_{train} \approx 13$  min per una sola categoria, in totale  $13 * 6 = 1 : 30h$  il tempo complessivo risulta

$$t_{perceptron} \approx 1 : 30 h$$

## 5. Conclusioni

Abbiamo notato che le curve precision-recall sono migliori in caso di categorie con un numero di documenti di test maggiore di 1000 e con la metà di documenti di test.

## Riferimenti bibliografici

- [1] N.Cristianini and J.Shawe-Taylor, An Introduction to Support Vector Machines and Other Kernel-based Learning Methods, Cambridge University Press, 2002
- [2] <https://nlp.stanford.edu/IR-book/html/htmledition/naive-bayes-text-classification-1.html>
- [3] <https://nlp.stanford.edu/IR-book/html/htmledition/the-bernoulli-model-1.html>
- [4] <https://github.com/Carlone92/Text-Classification-Reuters21578>

Figura 1. Perceptron precision-recall curve categoria *earn*

Il grafico (Fig 1) rappresenta la precision in funzione della recall. Notiamo che tra [0.0 e 0.45] la curva si mantiene praticamente costante ed uguale a 1, cioè il test ha predetto i documenti in maniera corretta. Da 0.45 in poi abbiamo un peggioramento delle prestazioni con un andamento inizialmente brusco per poi migliorare leggermente.