Intelligenza Artificiale
Corso di Laurea in Ingegneria Informatica
Univeristà degli Studi di Firenze

Naive Bayes Text Classification

Alessio Bonacchi
Sessione in Aprile
2021

Obiettivo

Lo scopo del progetto è quello di implementare le versioni naive Bayes degli algoritmi per l'apprendimento e classificazione del testo:Bernoulli e Multinomiale.

Il train e test dei classificatori è stato eseguito sul dataset *Large Movie Review Dataset* che contiene 25000 documenti di testo(sia per train che per test) rappresentanti delle recensioni positive e negative di film raccolte da *Imdb*.Il goal è proprio quello di riuscire a predire la classe della recensione.

Tools utilizzati

Al fine di realizzare il progetto sono state utilizzate le seguenti librerie:

Sklearn: utilizzato per realizzare il bag of words con la classe *CountVectorizer* e le matrici di confusione con *metrics*.Inoltre ho utilizzato *load_files* per caricare i dataset da locale.

nltk: utilizzato per la tokenization tramite $word-tokenize$,e lo stemming tramite *PorterStemmer*.Inoltre è stato necessario per importare un set di stopwords da escludere dai documenti in analisi.

NumPy: oltre a fornire le matrici sparse per contenere i diversi bag of words, è risultato necessario al fine di salvare tali matrici sotto forma di file *.npz* in modo da non dover fare il preprocessing sul dataset ad ogni run.

Implementazione

Preprocessing

Le operazioni di caricamento del dataset sono rese possibili dallo script *loadSet*,che sfrutta la funzione *load_files* di *sklearn*.

Il dataset così caricato viene convertito, tramite *setBagofWordsM* o *setBagofWordsM*, rispettivamente nel set di bag of words appropriato al classificatore in uso e, successivamente salvato grazie a *numpy.savez_compressed* al fine di non dover ricalcolare questi dati ogni volta.Rispettivamente i file *.npz* inclusi nel progetto rappresentano:

- **trainBernoulli,trainMultinomial**: contiene la matrice sparsa che rappresenta i documenti di train come bag of words.

- **testBernoulli,testMultinomial**: contiene la matrice sparsa che rappresenta i documenti di test come bag of words.

- **trainTarget,testTarget**: contengono rispettivamente il vettore dei target per il trainset e per il testset.

- **vectorizerVocabulary**: contiene il vocabolario ottenuto vettorizzando il dataset di train.

Classificazione

Le classi $BernoulliClassifier$ e $MultinomialClassifier$ si occupano della classificazione dei documenti secondo il metodo *Bernoulli e Multinomial naive Bayes*.Realizzano a livello pratico l'algoritmo definito in *McCallum Nigam,1998*.

Ho deciso di utilizzare le $log - probabilities$ al fine di ottenere una precisione maggiore e in modo da non eccedere il limite di floating precision massima di *Python*: a livello teorico non cambia niente,in quanto i calcoli sono i medesimi,tuttavia a livello informativo sono stati necessari per effettuare correttamente la classificazione.I documenti vengono classificati tramite le seguenti formule:

Multinomial:

$$P(c_k|X) = (\sum_{i=0}^{|V|} \log_{10} condProb[k][i]^{xt}) + \log_{10} Prior[k]$$

,dove c_k rappresenta la classe $k - esima$, $condProb[k][i]$ la probabilità di trovare la parola $i-esima$ del vocabolario in documento di classe k e $Prior[k]$ la a priori per la classe $k-esima$ e $|V|$ è la dimensione del vocabolario.

Bernoulli:

$$P(c_k|X) = (\sum_{i=0}^{|V|} \log_{10}(b_t * condProb[k][i] + (1 - b_t) * (1 - condProb[k][i]))) + \log_{10} Prior[k]$$

,dove b_t denota se la parola $i - esima$ è presente in X (il resto è come sopra)

Risultati

Entrambi gli algoritmi performano bene,riuscendo ad arrivare ad una accuracy superiore all'80%

Anche a livello di tempi di esecuzione la performance è piuttosto buona:*Bernoulli* impiega generalmente più tempo di *Multinomial*(dovuto al fatto che in fase di prediction si deve iterare su tutto il vocabolario e non solo sulle parole presenti nel documento come invece accade in *Multinomial*).
Tuttavia il tempo di training e testing si attesta per entrambi sotto i 2/3 minuti.

Di seguito, come da richiesta, riporto le matrici di confusione generate dall'esecuzione dei classificatori.

Bernoulli:

Listing 1: Bernoulli naive Bayes

```
Contingency matrix: (rows: examples of a given true class)
[[11135  1365]
 [ 3087  9413]]
Classification report
              precision    recall  f1-score   support

         neg       0.78      0.89      0.83     12500
         pos       0.87      0.75      0.81     12500

    accuracy                           0.82     25000
   macro avg       0.83      0.82      0.82     25000
weighted avg       0.83      0.82      0.82     25000
```

Multinomial:

Listing 2: Multinomial naive Bayes

```
Contingency matrix: (rows: examples of a given true class)
[[10963  1537]
 [ 2902  9598]]
Classification report
              precision    recall  f1-score   support

         neg       0.79      0.88      0.83     12500
         pos       0.86      0.77      0.81     12500

    accuracy                           0.82     25000
   macro avg       0.83      0.82      0.82     25000
weighted avg       0.83      0.82      0.82     25000
```

Bibliografia

[1] Large Movie Review Dataset, IMDb,
https://ai.stanford.edu/ amaas//data/sentiment/

[2] McCallum & Nigam 1998,
https://www.cs.cmu.edu/ knigam/papers/multinomial-aaaiws98.pdf