Elaborato Intelligenza Artificiale

Federico Palai

02/09/2017

Introduzione

L'obiettivo di questo elaborato era quello di confrontare le prestazione di due classificatori, Naïve Bayes e Decision Tree, su più dataset. Per fare questo è stato utilizzata la libreria Scikit-learn [1] per Python e, in particolare, le implementazioni fornite dei classificatori, oltre alla funzione per poter effettuare un plot dei risultati.

Rappresentazione dei risultati

Al fine di valutare la classificazione, si è optato per dividere il dataset in ingresso, in due parti: una prima parte utile al test, composta dal 50% dell'intero dataset e una seconda parte utile al training. Quest'ultima viene incrementata di volta in volta in scala logaritmica, partendo da un minimo del 10% (rispetto all'intero corpo di dati), per arrivare fino al 50%. Così facendo, si è potuto osservare il comportamento dei due classificatori all'aumentare del numero di esempi di training a cui sono sottoposti. Ovviamente la divisione del training e del test (ben lontana dalla canonica divisione in 80-20) non doveva servire a valutare l'efficacia della classificazione, bensì a sottolineare l'eventuale distanza tra i modelli. A tale proposito, i grafici risultanti riportano sull'asse delle ascisse, le varie percentuali di training che stiamo considerando, mentre sull'asse delle ordinate, l'errore compiuto dal classificatore sul test. Riguardo quest'ultimo punto, è bene specificare che l'errore considerato è in realtà la media dell'errore che il classificatore commette sulle diverse percentuali di training, lasciando invariata la dimensione, ma cambiano gli esempi considerati. Ovvero, per rendere il più imparziale possibile la scelta degli esempi da selezionare come train e come test, non solo si è ritenuto di dover effettuare una selezione random dei valori, ma anche di ripetere questa selezione per un numero di volte fissato (20) al fine di non incorrere in risultati straordinari dovuti a esempi particolari, casualmente selezionati. Oltre a questo, sul grafico è presente la deviazione standard calcolata sull'insieme di risultati, ottenuti dopo le 20 iterazioni, per la percentuale di dataset corrente.

Specifica sui classificatori

I classificatori presi in esame, come già detto, sono Naïve Bayes e Decision Tree. Per entrambi ci sono delle precisazioni da fare.

Per quanto riguarda Naïve Bayes, è stata scelta la variante Multinomiale (non volendo falsare il risultato a causa della binarizzazione effettuata da Bernoulli Naïve Bayes). Inoltre abbiamo lasciato l'opzione di default fornita da Scikit-learn, riguardo il Laplace Smoothing ($\alpha = 1.$), in modo da assegnare una probabilità diversa da zero a quelle parole che non appaiono negli esempi (questo risulta importante per quanto riguarda i dataset testuali). Mentre, per Decision Tree, si è deciso di non effettuare alcun tipo di pruning sull'albero. Anche per questo motivo (crescita non controllata dell'albero), le prestazioni in costo temporale di Decision Tree risultano molto più elevate rispetto a quelle di Multinomial NB.

Dati Sperimentali

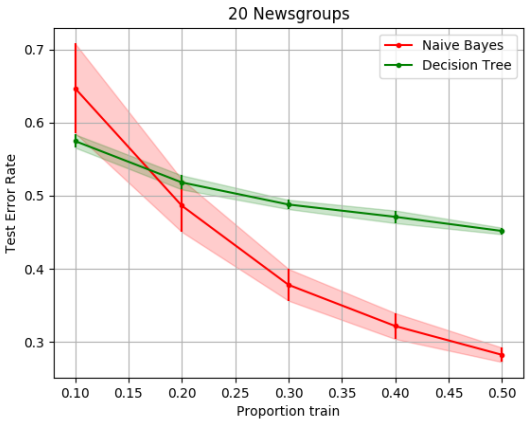
Di seguito, si riportano i risultati ottenuti mediante l'implementazione di uno script in Python, sulla base delle specifiche discusse fin'ora. Si suddivideranno quest'ultimi a seconda del dataset utilizzato. Per due dataset in particolare (*Twenty News Groups* e *Reuters-21578*), sono state applicate delle operazioni di preprocessing, in modo da adattare i dati per l'utilizzo del classificatore. Vediamo brevemente queste operazioni di preprocessing prima di mostrare i risultati:

- **CountVectorizer**: questa funzione, si preoccupa di convertire un insieme di documenti di testo in una matrice di conteggi di token e di creare un vocabolario, nel quale, ad ogni valore, è collegata la frequenza con cui appare nell'intero corpo.

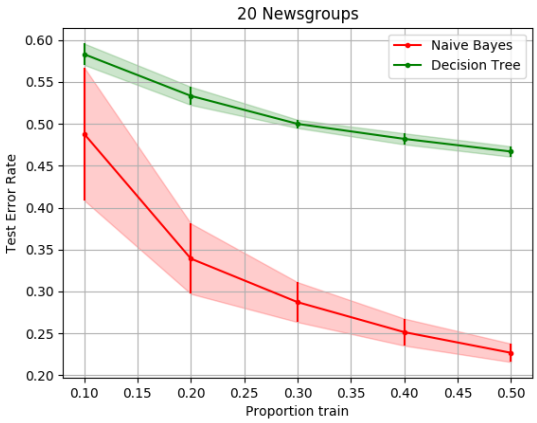
- **TfidfTransformer**: Si compone di due operazioni. La prima normalizza i valori di frequenza sulla base del numero totale di parole del documento (così che documenti lunghi non abbiano medie più alte di quelli corti), mentre la seconda scala il peso di quelle parole che compaiono in più documenti portando quindi un contenuto informativo più basso (queste parole sono dette *stop-words*).

1 Twenty News Groups [2]

Il primo dataset è *Twenty News Groups*, già presente nella libreria di Scikit-learn. Si tratta di un dataset testuale, composto da 18846 newsgroups, postati su 20 diversi topics. Come già detto, l'insieme di dati viene preprocessato prima di essere passato alla funzione di plotting: per prima cosa si elimina dal dataset *headers, footers* e *quotes*, per renderlo più realistico. Dopodiché si procede con le operazioni di preprocessing precedentemente descritte. Se l'operazione di `CountVectorizer` è necessaria, quella di `TfidfTransformer` non lo è. Distinguiamo allora i risultati ottenuti con e senza l'operazione:



(a) 20 News Groups senza trasformazione TFIDF



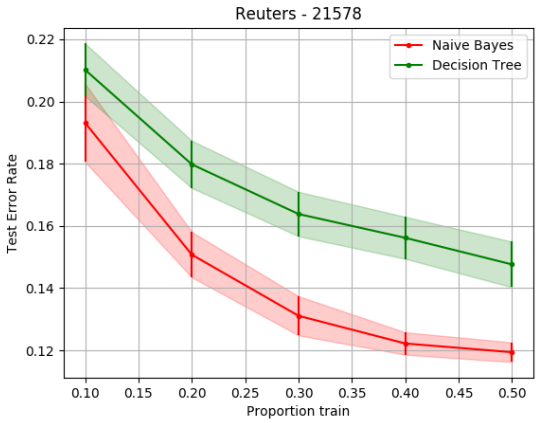
(b) 20 News Groups con trasformazione TFIDF

Notiamo che, nei due grafici, sebbene la classificazione fatta con `Decision Tree` non vari di molto (si passa da un errore inferiore al 60%, fino a uno vicino al 45%, per entrambi i grafici), quella relativa a `MultinomialNB` cambia. In particolare, l'errore è molto più alto quando non si effettua alcuna trasformazione (errore intorno al 65%), mentre si riduce quando questa viene effettuata (errore inferiore al 50%).

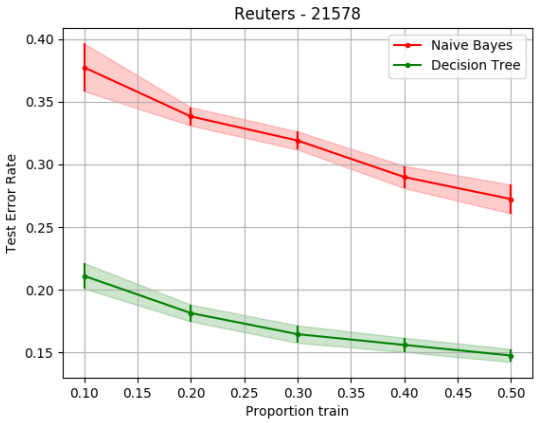
In generale, comunque, notiamo che: `MultinomialNB` raggiunge risultati migliori, sebbene con un'alta deviazione standard (è presente un'elevata variazione rispetto al valore medio), mentre `Decision Tree`, termina con un errore più alto, ma avendo una deviazione più contenuta (i valori di errore sono molto più vicini a quello di media).

2 Reuters-21578 [3]

Il dataset *Reuters-21578* considerato, è una variante ridotta del dataset originale. Considera infatti "solo" 21578 samples, contenuti in un numero ridotto di categorie (10 per l'esattezza), fra le principali e più popolate dell'intero dataset. Effettuiamo le stesse operazioni di preprocessing, già viste per *Twenty News Groups* e torniamo a distinguere sull'uso di `TfidfTransformer` . Otteniamo i seguenti risultati:



(c) Reuters - 21578 senza trasformazione TFIDF



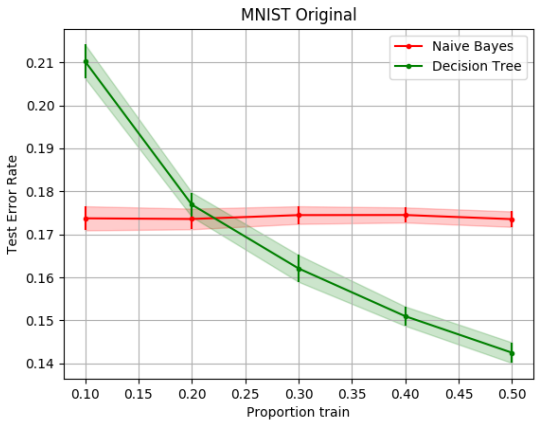
(d) Reuters - 21578 con trasformazione TFIDF

Al contrario di *Twenty News Groups*, per *Reuters - 21578*, non usare l'operazione di `TfidfTransformation` consente di ottenere dei risultati migliori per quanto riguarda `MultinomialNB`. Questo comportamento è forse da attribuirsi alla distribuzione fortemente non omogenea del dataset: infatti, analizzandolo, si può notare uno squilibrio nel numero di elementi presenti in ogni categoria. Questo peggioramento non è invece presente su `Decision Tree`, che rimane praticamente invariato.

Anche in questo caso, parlando in generale, notiamo che `MultinomialNB` ha una deviazione standard, nel caso si usi `TfidfTransformer`, più elevata rispetto al non utilizzo di quest'ultima. Inoltre questa varia al variare della percentuale di dataset (prima diminuisce, poi aumenta nuovamente dopo il 30%). `Decision Tree`, invece, mantiene una deviazione ridotta e costante in entrambi i grafici.

3 MNIST (Original) [4]

Il dataset *MNIST* è, diversamente dai precedenti, un dataset di tipo non testuale. Contiene infatti 70000 samples di immagini di dimensione 28×28 pixel, raffiguranti caratteri numerici scritti a mano. Le immagini sono suddivise in 10 classi. I risultati ottenuti con questo dataset sono i seguenti:

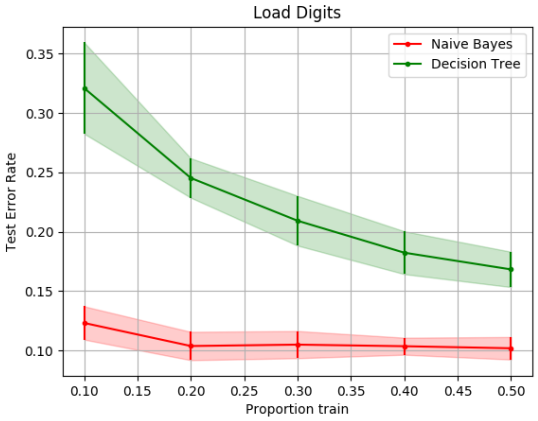


(e) MNIST Original

Su questo dataset possiamo notare un comportamento di `Decision Tree` nettamente migliore rispetto a `MultinomialNB`, all'aumentare del train set. Oltre a un comportamento peggiore, `MultinomialNB` mostra di non aumentare le sue capacità di predizione all'aumentare della dimensione del train set, anzi, a volte anche peggiorandole (come ad esempio fra il 20% e il 40%, nel nostro caso). La deviazione standard, per entrambi i classificatori, rimane pressoché costante.

4 Digits [5]

Anche questo dataset è di tipo non testuale, contenente sempre immagini raffiguranti caratteri numerici scritti a mano, ma in formato 8×8 pixel. Ha una dimensione di 1797 immagini, suddivise in 10 classi. É già presente nelle librerie di Scikit-Learn. Vediamo i risultati ottenuti:



(f) Load Digits

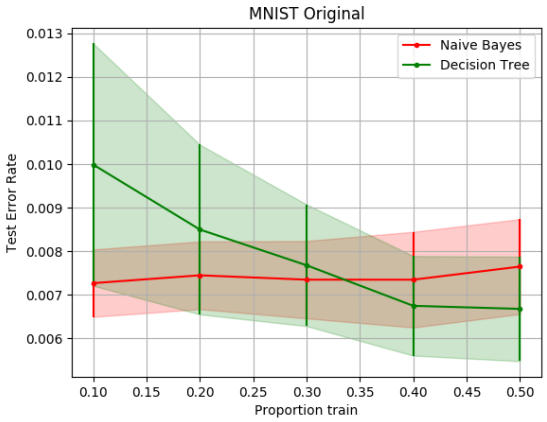
diversamente da *MNIST*, su questo dataset si nota una migliore prestazione di `MultinomialNB`, sebbene `Decision Tree` mantenga comunque un tasso di miglioramento al crescere della dimensione di train, più alto. `MultinomialNB` invece, migliora solo nell'aumento da 10% a 20%, rimanendo poi, pressoché costante. Anche in questo caso, la deviazione standard rimane pressoché costante.

Conclusioni

Riassumendo quanto riportato fin'ora, dal confronto fra questi due classificatori, possiamo notare come, generalmente, si ottengano migliori risultati con `MultinomialNB` piuttosto che con `Decision Tree` sebbene questo dipenda fortemente dai dataset utilizzati: sui due dataset testuali (*Twenty News Groups* e *Reuters - 21578*) infatti, abbiamo già discusso come la loro differenza influenzi il risultato, considerando due varianti per ognuno, con e senza l'uso di `TfidfTransformation`. Per quanto riguarda i dataset di immagini (*MNIST*, e *Digits*), invece, si ottiene un risultato di parità (meglio `Decision Tree` su *MNIST*, meglio `MultinomialNB` su *Digits*). Questa discordanza su due dataset molto simili, ritengo sia da attribuire a due fattori:

1. La tipologia di dati. In *Digits*, le immagini hanno grandezza 8×8 pixels, contro i 28×28 di *MNIST* e sono pertanto meno definite (e quindi più difficili da predirre).
2. La grandezza del dataset. *MNIST* può contare su 70000 esempi diversi, mentre *Digits* su appena 1797. Questo comporta un miglior allenamento dei classificatori che operano sul primo, mettendo a disposizione una dimensione del train set più grande.

In particolare il secondo punto, dimostra come la dimensione del dataset sia importante nella considerazione di `Decision Tree`. A prova di questo, ho provato a considerare una porzione più piccola di *MNIST*, passando da 70000 a 10000 dati. Il risultato è il seguente:



Notiamo che, nonostante si ottenga un errore praticamente ridotto a zero, `Decision Tree` rimane più performante, ma la distanza fra i due si è molto ridotta.

In conclusione, possiamo vedere come l'assunzione *naïve* di indipendenza fra i dati, alla base di Naïve Bayes, se da una parte porta in generale ad un buon comportamento, la maggior parte delle volte migliore, nel nostro

caso, a quello di `Decision Tree`, dall'altra condiziona fortemente il funzionamento del classificatore. Questo concetto è ben rappresentato dai grafici di *Twenty News Groups* e *Reuters - 21578*, nei quali il comportamento di `MultinomialNB` è instabile rispetto al preprocessing dei dati, diversamente da quello di `Decision Tree`.

Diversamente, `Decision Tree` è probabile soffra le ridotte dimensioni del train set (che ricordiamo non superano il 50% del dataset complessivo). A dimostrazione di questo, nell'unico caso in cui la dimensione complessiva consente di avere comunque un elevato numero di samples per il training (*MNIST*), `Decision Tree` ha un comportamento migliore.

Riferimenti bibliografici

[1] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[2] K. Lang, "Newsweeder: Learning to filter netnews," in *Proceedings of the Twelfth International Conference on Machine Learning*, pp. 331–339, 1995.

[3] D. D. Lewis, Y. Yang, T. Rose, , and F. Li, "Rcv1: A new benchmark collection for text categorization research," *Journal of Machine Learning Research*, vol. 5, pp. 361–397, 2004.

[4] Y. LeCun, L. Bottou, Y. Bengio, , and P. Haffner, "Gradient-based learning applied to document recognition.," *Proceedings of the IEEE*, vol. 86(11), pp. 2278–2324, November 1998.

[5] F. Alimoglu and E. Alpaydin, "Methods of combining multiple classifiers based on different representations for pen-based handwriting recognition," *Proceedings of the Fifth Turkish Artificial Intelligence and Artificial Neural Networks Symposium*, June 1996.