Relazione progetto Intelligenza Artificiale

di Stefano Agostini

1. Introduzione

L' elaborato assegnato richiedeva di studiare l'errore di generalizzazione in funzione del numero d'esempi applicando Naive Bayes e Decision Tree.
Si richiedeva quindi di usare il dataset di caratteri manoscritti MNIST e di confrontare su di esso le prestazioni dei due classificatori.

2. Implementazione

L'elaborato è stato svolto in Python, versione 3.6.3. Le dipendenze installate per eseguire il codice implementato sono:

- ScyPy, richiesto per l'installazione di scikit-learn;
- NumPy, richiesto per l'installazione di scikit-learn;
- Scikit-learn, contenente gli algoritmi necessari per i classificatori richiesti;
- PyPlot, per la visualizzazione dei grafici.

Tutte queste dipendenze sono state installate tramite il comando da terminale *pip install* seguito dal nome della dipendenza desiderata.
Il codice è in formato .ipynb, eseguibile tramite la web app Jupyter Notebook, che consente di condividere documenti che contengono live code.

2.1. Dataset

Per la manipolazione del dataset MNIST sono state utilizzate funzioni reperite nella documentazione di scikit-learn.
Per quanto riguarda il fetch del database, è stata usata la funzione fornita direttamente dalla libreria *scikit-learn* **fetch_mldata()** che ritorna i 70000 caratteri manoscritti, dei quali 10000 serviranno come test-set.

2.2. API utilizzate

L'implementazione si basa sui classificatori Naive Bayes e Decision Tree reperibili in *scikit-learn*. In particolare, è stato usato il classificatore BernoulliNB() per quanto riguarda l'implementazione di Naive Bayes, binarizzando i valori dei pixel delle immagini. Gli alberi di decisione, invece, sono stati implementati sia con l'indice di Gini che con l'entropia come misura della qualità dello split.
Per determinare i risultati da visualizzare sui grafici è stata usata la funzione **learning_curve()** di scikit-learn la quale ci ha fornito i risultati dei test al variare della dimensione del train-set.
ShuffleSplit() permette di creare una strategia di cross-vaidation per il learning supervisionato. Per mantenere gli stessi risultati in diverse esecuzioni è stato passato il parametro *random_state=0* alla funzione di creazione della strategia.
La funzione **plot_curve()**, la quale provvede a disegnare i grafici a partire dai dati generati dalla funzione learning_curve(), è stata ripresa da un esempio fornito in classe dal professore.

3. Riproduzione dei risultati

Per eseguire lo script bisogna avere installato la Jupyter Notebook. Da linea di comando, aprire la web app tramite il seguente comando:

jupyter notebook

All'esecuzione di tale istruzione, si aprirà il browser in una schermata dove sarà possibile ricercare la cartella del proprio computer nella quale è stato salvato il file (verosimilmente, per Windows, nella cartella *Downloads*). Fare doppio click sul file Progetto-esame.jpynb e eseguire le prime quattro sezioni in ordine per permettere di caricare le dipendenze e le funzioni richieste dalle sezioni successive, che potranno essere eseguite nell'ordine desiderato. Il file

contiene già un run eseguito per avere subito a disposizione i risultati senza dover attendere tempi mediamente lunghi (per Decision Tree, si tratta di almeno 15 minuti).

4. Risultati e conclusioni

4.1 Decision Tree

Dai grafici sotto riportati in Figura 1, si evidenzia che le due versioni dell'implementazione di Decision Tree comportano risultati simili: in entrambi, un aumento del numero di esempi porta ad un miglioramento del test-score, segno che il classificatore beneficia di tale aumento, arrivando ad avere con train-set di 60000 caratteri una percentuale di errore sul test-set attorno al 12%.

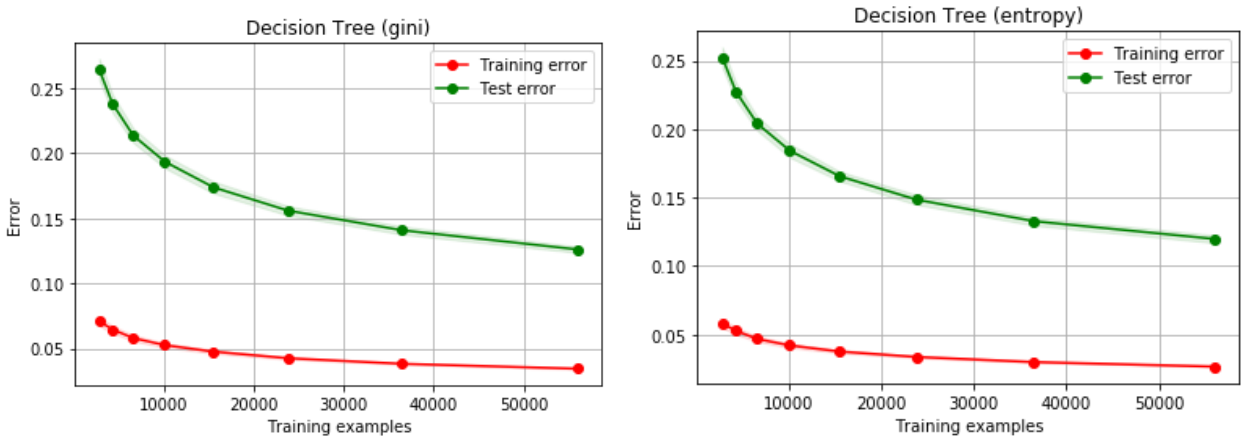

Fig. 1: Curve di apprendimento del classificatore Decision Tree

La differenza tra le due si nota maggiormente per quanto riguarda piccole dimensioni del train-set, sulle quali l'indice di Gini provoca un errore più alto rispetto all'uso dell'entropia come misura della qualità degli split. L'overfitting, in entrambi i casi, tende a diminuire notevolmente nei primi risultati con train-set bassi per poi andare quasi a stabilizzarsi quando i caratteri di apprendimento superano quota 30000.
Stranamente, rispetto a quanto si potrebbe aspettare, anche il train-score migliora all'aumentare del numero di esempi, facendo pensare ad una potatura dell'albero non troppo pesante.

4.2. Bernoulli Naive Bayes

Anche in questo caso, come possiamo notare dal grafico in Figura 2, l'errore sul test-set diminuisce all'aumentare degli esempi, anche se in modo meno visibile rispetto al caso dell'albero di decisione.

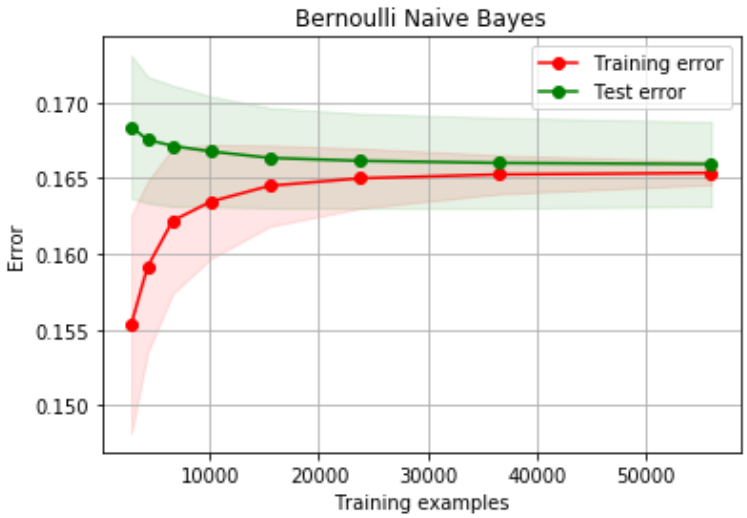

Fig.2: curva di apprendimento del classificatore Bernoulli Naive Bayes

L'overfitting arriva ad essere quasi nullo per grandi dimensioni del train-set e rimane quasi stabile sopra ai 20000 esempi. Si hanno grandi differenze di risultati al variare dei samples scelti per il learning supervisionato anche con insiemi di dimensioni uguali: in alcune parti del grafico, la deviazione standard degli score sul train-set e sul test-set fa sì che gli score migliori sul test-set siano più bassi degli score peggiori sul train-set. Notiamo però che il test-score non è molto elevato per insiemi non troppo grandi del test-set, indice che le assunzioni del modello di Bernoulli e la binarizzazione dei pixel delle immagini non influisce troppo sui risultati.

4.3. Conclusioni

Si nota che, per grandi dimensioni del train-set, Decision Tree è sicuramente il miglior algoritmo da usare tra i due qui analizzati. L'implementazione di Bernoulli Naive Bayes, però, può contare su una maggior velocità di calcolo: infatti, questo impiega poco più di due minuti per creare la curva di apprendimento, mentre l'albero di decisione impiega nel caso migliore almeno 15 minuti. Inoltre, Bernoulli provoca un test-score migliore di oltre il 5% nel caso di train-set non troppo estesi.

5. Bibliografia

- Stuart J. Russell, P. Norvig, Arti_cial Intelligence: A Modern Approach, 3rd Edition, Pearson Education, 2010.
- http://scikit-learn.org/stable/documentation.html, scikit-learn documentation.