Università degli Studi di Firenze

Elaborato di Intelligenza Artificiale: “Confronto tra algoritmi”

Matteo Bavecchi

Corso di Intelligenza Artificiale

Prof. Paolo Frasconi

Indice

Obiettivo

In questo esercizio si confrontano risultati sperimentali degli algoritmi Perceptron (Perc), Random Forest (RF), e Naive Bayes (NB) su una collezione di datasets seguendo la procedura sperimentale descritta da Zhang & Suganthan (2014) ed utilizzando implementazioni disponibili degli algoritmi di apprendimento (p.es. scikit-learn in Python o Weka in Java).
In particolare, si cerca di riprodurre alcuni risultati sperimentali seguendo lo schema della tabella 2, ma con colonne RF, Perc e NB (ovvero ignorando completamente PCA-RF, LDA-RF e RF-ensemble ed aggiungendo due classificatori diversi), e limitatamente a 6 datasets da scegliere a piacere tra quelli con almeno 2000 esempi. Si noti che non è richiesto comprendere o sperimentare con le tecniche descritte nella sezione 2 dell'articolo dal quale va solo presa la metodologia sperimentale di sezione 3.

Implementazione

Scelti i 6 datasets tra quelli usati nell'articolo, la maggior parte di questi sono stati partizionati con il metodo Repeated K-Fold Cross Validation. Uno di questi invece è stato partizionato prendendo una parte di dati per la fase di training e la restante per la fase di test.
Gli esperimenti sono stati ripetuti per 15 volte, ed è stata calcolata la media dell'accuratezza e la deviazione standard.

Per implementare l'esperimento sono stati usati i seguenti script Python:

config.py

Contiene i metodi **get_models()** e **execute(X,y)**, che servono rispettivamente a restituire i classificatori utilizzati nell'esperimento (Random Forest, Naive Bayes e Perceptron) e per calcolare l'accuratezza dei vari classificatori.
Il cross-validator generator utilizzato per tutti e 5 i datasets è **sklearn.RepeatedKFold**, configurato con 15 ripetizioni, 10 split e un random state di 50.
Il metodo **cross_val_score()** esegue la cross validation, facendo training e test in automatico. Restituisce quindi l'accuratezza della predizione.
Si esegue quindi un ciclo per eseguire tutti e tre i classificatori del vettore models.

kfold.py

Vengono caricati i dataset in file CSV e vengono manipolati per prepararli ad essere processati dagli algoritmi. Per molti vengono fatte minime modifiche, come isolare la classe per metterla nel vettore y. Il dataset Ozone invece ha avuto bisogno di più accortezze, come ad esempio la sostituzione dei caratteri "?" con il simbolo NaN e la successiva sostituzione con la media della colonna. Inoltre la prima colonna del dataset è stata eliminata, perché conteneva la data di misurazione.

letters.py

Contiene il codice riguardante l'unico dataset nel quale non è stato eseguito il Kfold cross validation, in quanto i dati sono già stati partizionati e bilanciati.
Il dataset viene scaricato dai database di OpenML.
Iterativamente, per 15 volte, si esegue il fitting e la predizione, e poi con la funzione **accuracy_score()** della libreria sklearn si stima l'accuratezza. Si stampa quindi la media e la deviazione standard dei 15 dati. Questo viene fatto per ogni algoritmo.

Datasets utilizzati

Per quanto riguarda la partizione dati:

- **15(15000,5000)** - Si esegue gli algoritmi 15 volte sul dataset diviso in 15000 istanze per il training e 5000 per il testing
- **15 10-fold cv** - Si effettua la partizione 10-fold cross validation ripetuta per 15 volte, quindi per ogni dataset si esegue l'algoritmo 150 volte

Un elenco dei datasets usati:

Dataset	Istanze	Attributi	Partizione dati
Letters	20000	16	15(15000,5000)
Nursery	12960	8	15 10-fold cv
Ozone	2536	72	15 10-fold cv
Page-blocks	5473	11	15 10-fold cv
WaveformV2	5000	40	15 10-fold cv
Wine quality (white)	4898	11	15 10-fold cv

Risultati

Dataset	Naive Bayes	Random Forest	Perceptron
Letters	64.04 ± 0.0	96.31 ± 0.13	37.42 ± 0.0
Nursery	73.3 ± 1.15	99.72 ± 0.15	78.30 ± 4.28
Ozone	69.21 ± 2.87	94.33 ± 1.33	91.41 ± 3.06
Page-blocks	90.11 ± 2.23	97.46 ± 0.63	94.33 ± 1.91
WaveformV2	80.01 ± 1.75	85.25 ± 1.52	80.69 ± 2.38
Wine quality (white)	44.32 ± 2.19	70.03 ± 1.97	41.64 ± 4.57

Tempi di esecuzione

Per velocizzare l'esecuzione dello script, è stato parallelizzato la computazione a n=8.
Computer utilizzato: Macbook air, CPU: Apple M1(ARM), 8 core
Tempo di esecuzione: 1:55:00

Conclusioni

Si nota che Random Forest risulta il più accurato tra gli algoritmi, e quello che ha la deviazione standard più bassa. Il suo svantaggio è la complessità, che ha un risvolto negativo sul tempo di training. Naive bayes è l'algoritmo più veloce dei tre, anche se è poco preciso.