Esercizio esame Intelligenza Artificiale

Leonardo Vannetti 5971442

Aprile 2021

1 Traccia dell' esercizio

In questo esercizio si utilizzano implementazioni disponibili di Decision Tree e Perceptron (p.es. scikit-learn in Python o Weka in Java) al problema dell'identificazione dello scriba su testi antichi, come descritto in
[De Stefano et al., 2018], cercando di riprodurre risultati analoghi a quelli riportati nella tabella 7 dell'articolo ma sostituendo DT con random forest e SVM con Perceptron, e tralasciando k-NN e NN (i risultati potranno quindi differire da quelli riportati nell'articolo).Il dataset è disponibile qui:

`http://archive.ics.uci.edu/ml/datasets/Avila`

È accettabile utilizzare un sottoinsieme dei dati se le risorse di calcolo disponibili sono insufficienti

2 Introduzione

Nello svolgimento dell'esercizio ho deciso di utilizzare Python con scikit-learn [Pedregosa et al., 2011] a differenza dell'articolo in questione [De Stefano et al., 2018] che utilizza Weka. I risultati da riprodurre, con le opportune modifiche agli schemi di classificazione, sono riportati in Tab.1. Per quanto riguarda il dataset, esso è stato utilizzato interamente e con nessuna modifica.

3 Scelta dei parametri

Analogamente all'articolo [De Stefano et al., 2018] ho utilizzato un approccio "grid-search", sui dati del training set, per trovare i parametri che meglio ottimizzassero i classificatori.

Tra le varie tuple di parametri provate sono state scelte quelle con la migliore 5-fold cross-validation accuracy. Differentemente dall'articolo sopra citato, nel quale è stata usata una 10-fold cross validation, ho preferito usare una 5-fold cross-validation perchè la classe con meno campioni ne ha solo 5 (sia nel training set che nel testing set) e ciò mi ha permesso di avere almeno 1 esempio per ogni classe durante la cross-validation.

A seguito entrerò più nello specifico per la scelta dei parametri per i due classificatori usati nell'esercizio.

Tabella 1
Precision e recall (espresse in percentuali), per ogni scriba, dei due schemi di classificazione.

Writer	DT		SVM	
	prec.	rec.	prec.	rec.
A	98.0	99.6	82.6	88.8
B	66.7	80.0	66.7	80.0
C	95.2	97.1	83	75.7
D	97.2	97.5	85.6	77.6
E	96.7	97.4	85.5	83.4
F	98.9	98.9	76.9	71.0
G	98.4	97.5	73.8	72.5
H	98.0	95	84.6	76.2
I	100	98.8	99.6	97.8
W	100	97.8	91.7	97.8
X	96.9	88.9	93.6	89.7
Y	100	98.9	90.5	88.8

Per **Random Forest** ho scelto i seguenti parametri:

- il minimo numero di campioni per foglia N_l
- il minimo decremento di impurità necessario per permettere lo split di un nodo I_d (per valutare l'efficacia del pre-pruning)
- il minimo numero di campioni di un nodo necessari per lo split N_t
- il parametro di complessità α, usato nell'algoritmo Minimal Cost-Complexity per il post-pruning (all'aumentare di α aumenta il post-pruning).

Gli insiemi dei valori testati sono [1, 2, 3, 4, 5] per N_l, [0, 0.01, 0.1] per I_d, [2, 3, 4, 5, 6] per N_t ed infine [0, 0.01, 0.015] per α. Il criterio per misurare l'impurità selezionato è l'entropia.

La tupla che ha massimizzato la 5-fold cross-validation accuracy è stata (N_l= 1, I_d= 0, N_t= 4, α= 0).

Questo risultato differisce da [De Stefano et al., 2018] soprattutto per quanto riguarda il post-pruning, che nel mio caso ($\alpha = 0$) non sembra portare alcun miglioramento, nonostante il noto problema di overfitting degli alberi di decisione.

Per confermare questo risultato ho effettuato un ulteriore esperimento per misurare come cambia l'accuracy di un singolo albero di decisione al variare del parametro α, sia sul training set che sul testing set. I risultati sono riportati in Fig.1.

Si può osservare come per un singolo albero la migliore accuracy sul testing set si ottiene proprio per $\alpha = 0$, e ciò si ripercuote sul classificatore random forest.

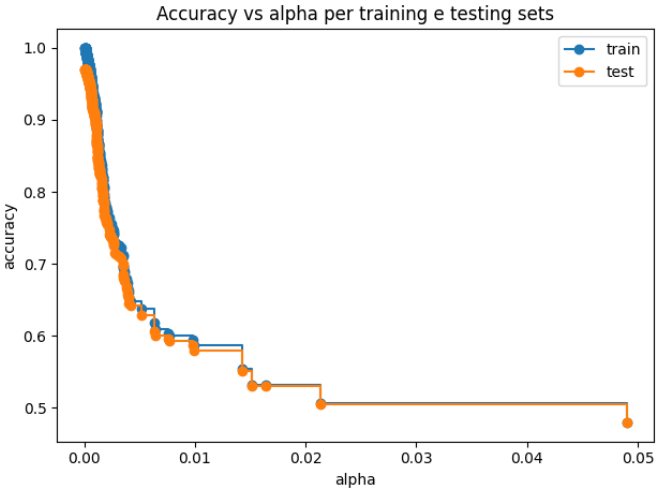

Figura 1: Andamento dell'accuracy al variare di α sul training e testing sets.

Per **Perceptron** ho scelto i seguenti parametri:

- Il massimo numero di iterazioni M_I
- Il peso associato ad ogni classe W_c, normalmente di valore 1, ma aggiustato in base alla frequenza della classe quando in modalità "balanced".

Gli insiemi dei valori testati sono [100, 1000, 10000] per M_I e [None, 'balanced'] per W_c.

La coppia che ha massimizzato la 5-fold cross-validation accuracy è stata ($M_I = 100$, $W_c = None$).

Il basso valore di M_I può voler dire che perceptron ha trovato velocemente un iperpiano separatore del training set oppure che i dati non sono linearmente separabili ed ha già iniziato ad oscillare, non trovando nessun iperpiano migliore in epoche successive. Vedremo poi dai risultati sperimentali riportati in Sez.4 che quest'ultima è l'ipotesi giusta.

Per quanto riguarda W_c tenere conto della frequenza delle classi non ha comportato nessun miglioramento, nonostante il dataset sia fortemente sbilanciato.

Tabella 2
Precision e recall (espresse in percentuali), per ogni scriba, dei due schemi di classificazione.

Writer	RF		Perceptron	
	prec.	rec.	prec.	rec.
A	99.3	99.7	54.2	86.5
B	100	100	0	0
C	97.1	98.1	100	1.0
D	100	97.5	12.0	0.8
E	97.9	99.3	29.4	30.0
F	99.6	99.3	6.7	0.6
G	99.8	98.9	12.8	9.2
H	99.6	99.0	4.1	3.5
I	100	99.6	69.6	77.0
W	100	100	0	0
X	99.4	98.1	82.1	52.7
Y	100	99.6	54.7	56.9

4 Risultati sperimentali e conclusioni

I risultati ottenuti sul testing set dai due classificatori proposti nell'esercizio (Random forest e Perceptron), usando i parametri descritti in Sez.3, sono riportati in Tab.2.

Confrontati con i risultati della Tab.1, sia Random Forest che Perceptron si comportano come mi aspettavo e in accordo con la teoria:

- **Random Forest** performa meglio di Decision Tree su ogni scriba, ottenendo ottimi risultati anche sulle classi con meno esempi.
- **Perceptron** performa molto peggio di SVM, ottenendo risultati per nulla soddisfacenti.

  Com'era intuibile il dataset non è linearmente separabile (molti esempi a bassa dimensione), di conseguenza nessun classificatore lineare potrà ottenere buoni risultati.

  Nonostante ciò in [De Stefano et al., 2018] SVM, che comunque si basa sul concetto di separazione lineare, ha ottenuto risultati nettamente superiori.

  Questo è dovuto al fatto che è stata usata la funzione kernel RBF, adattando meglio il classificatore per dataset non linearmente separabili.

Riferimenti bibliografici

[De Stefano et al., 2018] De Stefano, C., Maniaci, M., Fontanella, F., and Scotto di Freca, A. (2018). Reliable writer identification in medieval manuscripts through page layout features: The "avila" bible case. *Engineering Applications of Artificial Intelligence*, 72:99–110.

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