Confronto fra Decision Tree e Perceptron

Giovanni Burbi

February 2018

1 Introduzione

L'esercizio consiste nel confrontare le curve di apprendimento ottenute usando due algoritmi di apprendimento: Decision tree e il Perceptron. Questi algoritmi vengono applicati su un dataset di immagini che rappresentano dei prodotti di Zalando: Ci sono 60000 esempi di train e 10000 esempi di test, ognuno con 10 tipologie diverse di prodotti.

Si vuole confrontare le curve di apprendimento dei due algoritmi applicati al riconoscimento degli articolo del dataset.

Sarà mostrato che decision tree usando l'entropia come criterio di misurazione di impurità permette una percentuale di errori nel training inferiore rispetto al perceptron.

2 Descrizione del codice

Il codice consiste di 2 file .py:

il primo, Functions.py, contiene tutte le funzioni necessarie per ottenere i risultati che servono per il confronto fra gli alberi di decisione e il perceptron:

- Una funzione Plot necessaria per mostrare il grafico che rappresenta la curva di apprendimento di un classificatore. Riceve come parametri i valori restituiti dalla funzione $learning_curve$ presente nella libreria sklearn e un titolo con una descrizione opzionale da inserire nel grafico.

- Una funzione Parameters che serve a definire i parametri della funzione di cross validation e il numero di esempi che verranno generati e usati per costruire il grafico della curva di apprendimento. In questo esercizio viene usata la ShuffleSplit della libreria sklearn per effettuare la cross validation

- Due funzioni,PlotDecisionTree e PlotPerceptron, per plottare le curve di apprendimento del perceptron e del decision tree, prendono in ingresso la funzione di cross validation selezionata, il train sizes, che fissa su base logaritmica il numero di campione del training usati per il grafico, il set di train e il set di test divisi entrambi in input (X) e output (y) e i parametri

necessari per la funzione sklearn che implementa il classificatore. Le funzioni fornisco lo score del training, la accuratezza della predizione sul set di test, la confusion matrix e la classification report di sklearn. Queste informazioni consentono di valutare le prestazioni del classificatore.

- Altre due funzioni, DecisionTreeGridSearche PerceptronGridSearch, che effettuano la grid search usando la funzione sklearn gridsearchcv, queste funzioni prendono come parametri la funzione per la cross validation e il training set. Restituisce il miglior set di parametri, il miglior score fra tutte le combinazione dei parametri scelti all'interno della funzione e lo score sul training set di ogni combinazione di parametri con relativa imprecisione.

- L'ultima funzione, BestEstimatorCurve, riceve come parametri il tipo di classificatore, i set di train e test, il train sizes e la funzione di cross validation. Essa effettua la ricerca dei parametri migliori per il classificatore scelto e li usa per plottare la learning curve corrispettiva. Restituisce anche la accuratezza della fase di training, la accuratezza della predizione sul test set, la confusion matrix e la classification report per il classificatore usando i parametri che restituiscono il miglior score nel training.

Nel secondo file, main.py, viene caricato il data set e posto nei vettori che definisco l'input e l'output sia per il training set che per il test set, successivamente vengono chiamate le funzioni del file Functions.py per ottenere i risultati.

3 Descrizione del lavoro e risultati sperimentali

Nel main, il passo iniziale consiste nel caricare il data set e metterlo nei vettori di train e test, di ognuno di essi è costituito da input e output.

Dopo aver definito la funzione di cross validation e il train sizes, chiamando la funzione Parameters, viene effetuata la chiamata a BestEstimatorCurve sia per il decision tree che per il perceptron in modo da ottenere i grafici in *figura 1* e *figura 2* che rappresentano la curva di apprendimento per la miglior combinazione di parametri dei due classificatori.

Per il perceptron vengono provati vari valore del parametro maxIter e viene scelto quello per cui si ha la accuratezza migliore nel processo di training. Per il decision tree invece vengono variati il valore di maxDepth e il tipo di criterio di misurazione dell'impurità, nel nostro caso entropia e gini. La funzione applicata al decision tree restituisce la migliore accuratezza nel training, pari al 81.65% utilizzando l'entropia come criterio di misurazione dell'impurità e profondità dell'albero pari a 12. Per il perceptron il valore del maxIter che genera il migliore score nel training è 28, a cui corrisponde una accuratezza del 79.60%. Con questi parametri viene predetto l'output sul set di test e vengono effettuate varie misurazioni:

Figure 1: Curva di apprendimento Decision Tree usando i migliori parametri calcolati

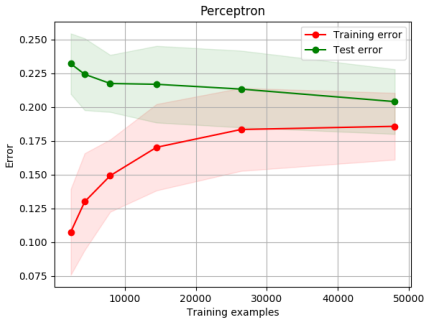

Figure 2: Curva di apprendimento Perceptron usando i migliori parametri calcolati

- Per il decision tree l'accuratezza della predizione è del 81.28% , la corrispettiva confusion matrix e classification report:

Classification report:

	precision	recall	f1-score	support
0	0.74	0.80	0.77	1000
1	0.97	0.94	0.95	1000
2	0.69	0.72	0.70	1000
3	0.82	0.81	0.82	1000
4	0.68	0.73	0.70	1000
5	0.92	0.89	0.91	1000
6	0.63	0.50	0.56	1000
7	0.87	0.89	0.88	1000
8	0.92	0.93	0.93	1000
9	0.90	0.91	0.91	1000
avg / total	0.81	0.81	0.81	10000

confusion matrix :

```
[[803   4  15  46   9   3 107   1  11   1]
 [ 10 938   6  32   5   0   7   0   2   0]
 [ 16   0 716  11 155   1  87   0  13   1]
 [ 55  22  17 811  67   1  18   0   6   3]
 [  9   1 144  43 727   1  70   0   5   0]
 [  0   1   0   2   0 893   1  64   6  33]
 [191   4 136  39  99   1 502   0  27   1]
 [  0   0   0   0   0  42   0 893   4  61]
 [  6   0  10   5   9  14  11  10 931   4]
 [  1   0   1   1   0  17   0  62   4 914]]
```

La prima mostra la precisione nel predire le varie classi e il numero di esempi, support, presenti nel test per ogni classe. La seconda mostra sulla diagonale quanti esempi sono stati predetti correttamente per ogni classe e sulla restante parte della matrice c'è il numero di volte che un campione è stato predetto erroneamente come un campione di un altra classe.

- Per il perceptron l'accuratezza della predizione è del 79.76% , la corrispettiva classification report e confusion matrix:

Classification report:

	precision	recall	f1-score	support
0	0.68	0.88	0.77	1000
1	0.96	0.95	0.96	1000
2	0.57	0.85	0.68	1000
3	0.78	0.86	0.82	1000
4	0.80	0.49	0.61	1000
5	0.92	0.89	0.91	1000
6	0.62	0.45	0.52	1000
7	0.91	0.93	0.92	1000
8	0.97	0.73	0.84	1000
9	0.92	0.94	0.93	1000
avg / total	0.81	0.80	0.79	10000

confusion matrix :

```
[[882   4  20  42   6   0  38   0   7   1]
 [  8 951   8  27   1   0   3   1   0   1]
 [ 27   6 854  22  45   0  45   1   0   0]
 [ 73  15  15 864  12   0  19   0   2   0]
 [  7   5 332  79 488   0  89   0   0   0]
 [  3   2   1   1   0 891   1  48   5  48]
 [245   3 194  52  52   0 445   0   8   1]
 [  0   0   0   0   0  40   0 930   0  30]
 [ 49   1  83  27   9  15  77   5 733   1]
 [  0   0   2   0   0  22   1  37   0 938]]
```

Successivamente vengono generate le curve di apprendimento per parametri non ottimali per vedere come si comporta il classificare in quei casi. Da *figura 3* si vede che l'albero di decisione con il limite di profondità settato ad un valore basso non riesce a classificare molto accuratamente i dati, infatti la accuratezza è del 71,5% sul train. Invece il perceptron in *figura 4* mostra che anche con poche iterazioni riesce a comportarsi bene raggiungendo una accuratezza del 78,3% sul train.

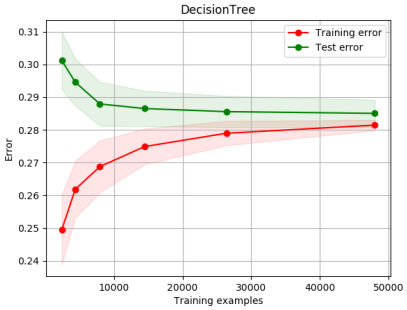

Figure 3: Curva di apprendimento Decision Tree usando entropy e con profondita massima di 5

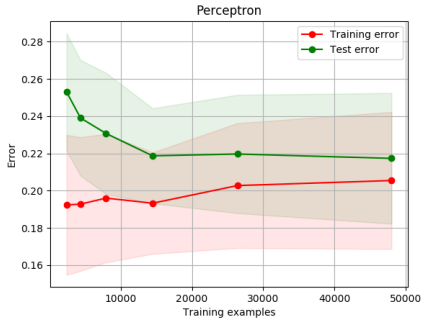

Figure 4: Curva di apprendimento Perceptron con iterazioni massime di 5

Nella *figura 5* il decision tree alla profondità di 20 mostra segni di overfitting, infatti l'errore sul training va a zero, l'accuratezza del training è del 96,8% . In *figura 6* il perceptron anche con 50 iterazioni rimane piuttosto costante come accuratezza e la varianza è notevolmente ridotta rispetto alla curva di apprendimento generata con meno iterazioni. (A causa delle limitate risorse dell'elaboratore utilizzato non è stato possibile andare oltre le 50 iterazione.)

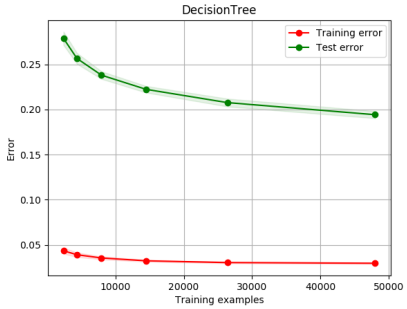

Figure 5: Curva di apprendimento Decision Tree usando entropy e con profondita massima di 20

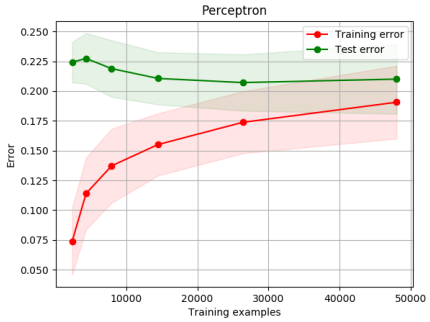

Figure 6: Curva di apprendimento Perceptron con iterazioni massime di 50

Di seguito sono riportati i valori dell'accuratezza del training per ogni parametro dei due classificatori:

Table 1:

max iter	accuracy
1	0.772 (+/-0.073)
2	0.780 (+/-0.061)
3	0.787 (+/-0.059)
4	0.786 (+/-0.054)
5	0.783 (+/-0.070)
6	0.792 (+/-0.049)
7	0.788 (+/-0.052)
8	0.789 (+/-0.058)
9	0.793 (+/-0.057)
10	0.790 (+/-0.062)
11	0.791 (+/-0.050)
12	0.787 (+/-0.064)
13	0.794 (+/-0.046)
14	0.791 (+/-0.064)
15	0.792 (+/-0.054)
16	0.792 (+/-0.058)
17	0.791 (+/-0.051)
18	0.790 (+/-0.059)
19	0.795 (+/-0.052)
20	0.789 (+/-0.066)
21	0.790 (+/-0.064)
22	0.790 (+/-0.061)
23	0.786 (+/-0.060)
24	0.794 (+/-0.044)
25	0.791 (+/-0.064)
26	0.793 (+/-0.053)
27	0.791 (+/-0.062)
28	0.796 (+/-0.048)
29	0.794 (+/-0.053)
30	0.788 (+/-0.067)
31	0.793 (+/-0.058)
32	0.786 (+/-0.062)
33	0.793 (+/-0.056)
34	0.789 (+/-0.057)

Table 2:

max depth	entropy accuracy	gini accuracy
1	0.197 (+/-0.006)	0.196 (+/-0.006)
2	0.357 (+/-0.009)	0.342 (+/-0.007)
3	0.503 (+/-0.009)	0.517 (+/-0.015)
4	0.651 (+/-0.011)	0.677 (+/-0.008)
5	0.708 (+/-0.007)	0.715 (+/-0.008)
6	0.733 (+/-0.007)	0.740 (+/-0.015
7	0.766 (+/-0.009)	0.772 (+/-0.010)
8	0.787 (+/-0.008)	0.790 (+/-0.009)
9	0.801 (+/-0.008)	0.803 (+/-0.008)
10	0.810 (+/-0.007)	0.811 (+/-0.007)
11	0.813 (+/-0.007)	0.815 (+/-0.007)
12	0.816 (+/-0.006)	0.817 (+/-0.007)
13	0.816 (+/-0.006)	0.816 (+/-0.007)
14	0.815 (+/-0.007)	0.815 (+/-0.007)