Progetto per IA
PerceptronVoted vs Perceptron

Fabian Greavu

03 Giugno 2018

1 Abstract

In questo articolo andrò ad applicare gli algoritmi di Perceptron e Perceptron Voted a quattro diversi dataset (uno linearmente separabile e gli altri no) per poi visualizzare i risultati raccolti e testare l'efficacia del Perceptron Voted a confronto con quelli del Perceptron. Progetto disponibile su [github]

2 Introduzione

Il perceptron appartiene alla classe degli algoritmi di supervised learning. Ideato da Frank Rosenblatt negli anni '60, fu una grande svolta nell'ambiente tanto da dare una svolta e incoraggiando la ricerca nel settore. Si basa sul concetto di un p̈ercettone,̈ o neurone al quale arrivano dei dati in ingresso e in uscita dà un risultato positivo o negativo.
Usandolo come classificatore binario dunque esso è in grado di effettuare l'apprendimento su un dataset in ingresso e poi predire un nuovo elemento. Se i dati sono linearmente separabili al 100%, allora avrà una accuratezza massima (100%). Invece (come è per la maggior parte dei casi) il dataset di train non è linearmente separabile e si cerca di massimizzare l'accuratezza (minimizzare l'errore). Testeremo i dataset usando sia il Perceptron che il VotedPerceptron, una versione in cui si tiene in memoria tutti i vettori dei pesi passati in modo da fare previsioni più accurate rispetto al Perceptron normale.

3 Datasets

In questo progetto utilizzeremo tre dataset di cui il primo linearmente separabile generato casualmente e gli altri disponibili online all'UCI Machine Learning Repository:

- HTRU2
- Occupancy Detection
- Banknote Authentication

4 Organizzazione

Per avere una versatilità maggiore tutti i file costruiti sono generici e si possono riutilizzare per nuovi test.
I due tipi di Perceptron sono implementati come due classi diverse (Perceptron, PerceptronVoted) derivanti da una astratta "PerceptronAbstract" che espone i metodi generali di uso (train, predict, getweights).
Tutti i datasets sono salvati nella cartella omonima 'datasets', così come i risultati sono salvati nella cartella "results".
Il file **Trainer.py** esegue il train e il test su ogni dataset. Si possono modificare le variabili interne in modo da effettuare test diversi:

- **ephocs**: un vettore avente i numeri di epoche da testare.
- **train_n**: il numero di esempi da fornire all'algoritmo per effettuare il training
- **validation_n**: il numero di esempi usati per effettuare la fase di validation sui dati appena addestrati
- **test_n**: il numero totale di esempi usati per predire e calcolare l'accuratezza del nostro perceptron
- **perceptron_types**: un vettore di dictionary contenete le informazioni delle classi da usare, dei loro nomi e identificatori. Volendo testare un nuovo algoritmo di apprendimento, basterà creare la classe apposita, derivarla da PerceptronAbstract e inserirla nell'apposito vettore

- **datasets**: non è una variabile globale, ma viene creata nel metodo $load\_datasets\_info$. E' un vettore di dictionary contenente le informazioni dei vari dataset, inclusa la loro posizione in memoria e il metodo da chiamare per inportare i dati dentro DatasetsFactory

Il file **ResultsViewer.py** invece, si preoccupa di prendere i risultati (cartella results) e plottare i dati. Esso viene automaticamente chiamato alla fine dell'esecuzione di Trainer.py ma si può chiamare separatamente solo per visualizzare gli ultimi risultati calcolati.
Se si usano dataset nuovi, modificare l'array **data** al suo interno come è descritto nel file.
Il file **DatasetsFactory.py** si occupa di fornire metodi statici per il prelievo dei dati dalle loro cartelle. Per ogni dataset c'è un metodo apposito di lettura (chiaramente varia se il file è csv, txt, ...).

5 Training

Il training è stato effettuato settando le variabili nel seguente modo:

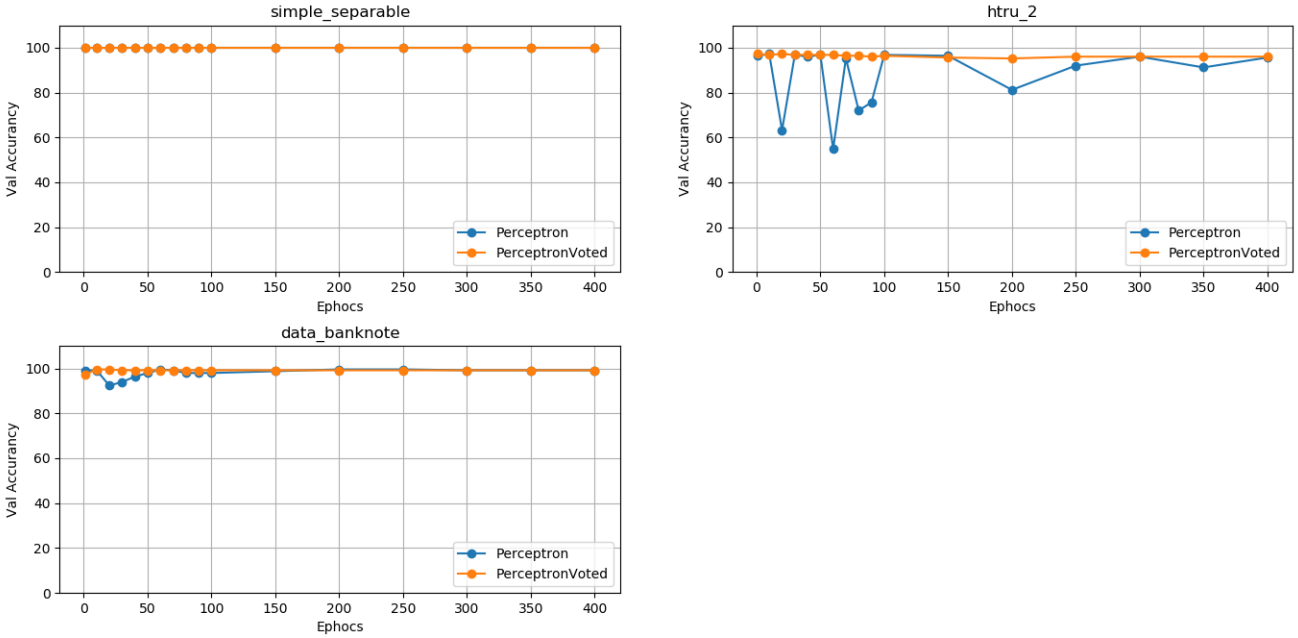
- train_n 500
- validation_n 250
- test_n 250
- ephocs [1, 5, 10, 15, ... , 95, 100, 150, 200, 250, 300, 350, 400]

Il numero di train, validation e test è stato scelto così perchè i dataset dovevano avere da esercizio almeno 1000 istanze e uno di essi ne aveva solo 1372. Abbiamo dunque seguito il modello 50/25/25 su 1000 istanze totali.
Ogni volta che si estraggono un numero di esempi da un dataset, essi vengono mischiati attraverso la funzione random passando un numero affinchè i test successivi vengano effettuati nello lo stesso ordine.
Il training è stato effettuato e tutti i risultati sono stati salvati nella cartella 'results' da cui possono essere disegnati i grafici.

6 Risultati

I risultati sono plottati su quattro grafici che mostrano l'accuratezza del validation test sia per il Perceptron che per il Perceptron Voted. Il plot è

caratterizzato da vari subplot riguardanti un certo dataset. Sulle ascisse vengono segnate le epoche, sulle ordinate l'accuratezza del validation per epoca. Di seguito allego il plot totale.



Ed ecco i risultati presi dal file All.txt nella cartella results per quanto riguarda il testing:

Table 1: Results.

Name	**Type**	**ephoc**	**Validation**	**Testing**
simple_separable	Perceptron	400	100.0%	100.0%
simple_separable	PerceptronVoted	400	100.0%	100.0%
htru_2	Perceptron	400	95.6%	95.6%
htru_2	PerceptronVoted	400	96.0%	96.4%
data_banknote	Perceptron	400	99.2%	98.0%
data_banknote	PerceptronVoted	400	99.2%	98.0%

7 Conclusioni

I risultati ottenuti rispecchiano le ipotesi formulate sui due tipi di algoritmi: per ogni dataset si ottiene una accuratezza maggiore usando il Perceptron Voted che usando il Perceptron base. Più precisamente osservando il crescere delle epoche si nota che la versione Voted converge più velocemente verso una accuratezza che tende al 100% mentre la versione normale ci impiega molte più epoche a stabilizzare la sua accuratezza (il dataset 'htru2' ha addirittura percentuali di accuratezza che oscillano tremendamente fino a 100 epoche mentre il 'banknote authetication' ci impiega circa 50 epoche). Queste percentuali affiorano dalla caratteristica dei dataset di essere più o meno linearmente separabili.
Per il dataset linearmente separabile si ottiene subito il 100% di accuratezza quindi entrambi i Perceptron funzionano bene.

8 Referenze

Publications:

- *- Yoav Freund , Robert E. Schapire, Large margin classification using the perceptron algorithm, Proceedings of the eleventh annual conference on Computational learning theory, p.209-217, July 24-26, 1998, Madison, Wisconsin, United States*
- *F. Rosenblatt, The Perceptron: A Probabilistic Model for Information Storage and Organization in The Brain, Psychological Review, y.1958, p.65-386*

Working class book:

- *Peter Norvig, Stuart J. Russell, Artificial Intelligence: A Modern Approach, ch.18, p.736-744*

Datasets from UC Irvine Machine Learning Repository:

- *https://archive.ics.uci.edu/ml/datasets/HTRU2*
- *https://archive.ics.uci.edu/ml/datasets/Occupancy+Detection+*
- *https://archive.ics.uci.edu/ml/datasets/banknote+authentication*