Perceptron e Voted Perceptron: confronto e applicazione

Riccardo Giachin

13 Giugno 2018

Abstract

In questo report del progetto spiego come ho applicato gli algoritmi di Perceptron e Voted Perceptron. Gli algoritmi saranno studiati su 3 dataset linearmente separabili e poi visualizzerò i risultati ottenuti facendo un confronto tra i due algoritmi.

1 Introduzione

Il *Perceptron*, in **machine learning**, è un algoritmo per il *supervised learning* dei classificatori binari. E' un tipo di classificatore lineare, che si basa su una funzione lineare di predizione che combina un set di pesi con un vettore caratteristico. Questo algoritmo fu creato da *Frank Rosenblatt* nel 1957 nel laboratorio aeronautico di Cornell per riconoscere le immagini che gli venivano presentate. Il Perceptron, in qualità di classificatore binario, è in grado di allenarsi su un dataset in ingresso e poi prevedere un nuovo elemento. L'accuratezza di tale algoritmo dipende dal dataset presentato e dal suo grado di separabilità. Nel caso in cui esso non sia separabile, l'algoritmo cercherà di massimizzare l'accuratezza. Il *Voted Perceptron*, sviluppato dalla mente di **Vladimir Naumovic Vapnik** è basato sul Perceptron di Rosenblatt, prende vantaggio dai dataset che sono linearmente separabili con un grande margine. Questo metodo è più semplice da implementare e anche molto più efficiente .

2 Accenni teorici

2.1 Perceptron

il Perceptron si basa sull'insegnamento di un classificatore binario, una funzione che mappa l'input x in un valore *f(x)* in modo tale che :

$$y = \begin{cases} 1 & \text{se} \quad w * x + b > 0 \\ 0 & \text{se} \quad otherwise \end{cases}$$

dove w*x è il dot product

$$\sum_{i=0}^{m} w_i * x_i$$

m è il numero di input e b è il fattore di Bayes. Per ciascun esempio x sarà fatta l'operazione descritta sopra, nel caso in cui y non sia concorde allora dovremmo aggiornare i coefficienti di w e b attraverso le operazioni w = w+y*x e b=b+y. Nell'altra eventualità si potrà passare all'input successivo poiché la classificazione è risultata secondo quel piano.

2.2 Voted Perceptron

Questo tipo di Perceptron permette di sopperire a quelle criticità che non sono risolvibili con il Perceptron standard. Infatti, nel caso in cui i dati non siano linermente separabili, il tipo standard genererà sempre un piano diverso se eseguissimo un controllo per limitare le situazioni. Usiamo, a questo scopo, il numero di classificazioni corrette che, nella fase di testing, determinerà il segno di un esempio secondo l'equazione:

$$\sum_{i=0}^{k} c_i * sign(w * x)$$

3 Datasets

In questo progetto vengono utilizzati tre dataset linearmente separabili disponibili online sul sito di UCI Machine Learning Repository:

	n Oggetti	n Train	n Test
Poker Hands	1025010	768758	256252
Air Quality	9358	7019	2339
Dota 2	102944	75000	27944

Il primo dataset ci offre un set di 5 carte scelte casualmente e, grazie al perceptron, calcoliamo l'accuratezza con la quale possiamo trovare una delle 9 possibili scelte tra quelle del poker classico.
Il Dataset AirQuality campiona la quantità di materiali presenti nell'aria a seconda della data e del tempo nel quale si trova.
Dota2 è già diviso in train e test, senza necessità di separarlo attraverso l'utilizzo della classe Scikit-learn; questa descrive vari risultati delle singole partite che vengono giocate. In tal caso il programma cerca di prevedere con quale accuratezza un singolo personaggio possa vincere una partita. I dati sono classificati secondo i nomi dei champions.

4 Classi

In questa sezione verranno descritte le classi progettate in questo lavoro.

4.1 Repositories

In questa sezione vengono elencate le Repos che ho utilizzato al fine di poter utilizzare il programma.

- **Matplotlib**: libreria utile per poter costruire e disegnare un grafico. Utilizzata per poter visualizzare i risultati dell'accuratezza.
- **Numpy**: utilizzata per risolvere i casi di moltiplicazioni tra vettori ed altri funzioni che supportano la libreria Sklearn.
- **Pandas**: permette di leggere i file esterni quali .csv e .excel.
- **sklearn**: permette di dividere un file in due: Train e Test, inoltre ha funzioni importanti quali calcolare la matrice di confusione, l'accuratezza e il report di classificazione.

4.2 Program's Class

- **Datasets**: questa classe contiene i vari file.csv, che andremo poi a studiare nelle altri classi.
- **Perceptron**: vengono implementati, oltre al costruttore, i metodi di training e di guess che permettono: il primo di allenare il nostro oggetto, mentre il secondo ci permette di poter provare a indovinare quale sarà il prossimo output del nostro perceptron.
- **Voted Perceptron**: come suggerisce il nome, vengono implementati i metodi del Voted perceptron. Anche in questo caso vengono implementati i metodi di training e di guess che implementano e sviluppano i metodi del Perceptron.
- **Air Quality Test**: viene creato il metodo che permette di testare gli algoritmi di Perceptron e Voted Perceptron sul dataset di AirQuality.
- **PokerHandTest**: studia gli algoritmi sul dataset pokerHand.csv
- **DotaTest**: in questo caso, avendo sia la classe test che train, non ho dovuto utilizzare il metodo di scikit-learn per dividere il dataset in train e test.
- **Grafic**: questa classe è composta da 3 metodi che permettono di disegnare un grafico, differente per ogni dataset, che dimostra la differenza di accuratezza tra i due algoritmi.
- **AllTest**: classe che racchiude tutte le precedenti classi e permette di poter richiamare un metodo senza dover andare a dover fare una run di ogni singola classe.

5 Risultati finali

5.1 Grafici

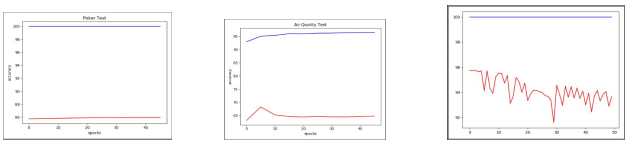

Figure 1: immagini relative ai grafici di differenza di accuratezza, dove la funzione del Perceptron

In questo caso possiamo facilmente vedere che con il perceptron(colore rosso) gli errori possono sempre presentarsi ad ogni iterazione. Mentre nel caso del Voted perceptron l'accuratezza è quasi sempre massima, dovuta al miglioramento dell'algoritmo base da parte di Vipnik

5.2 Matrice di confusione

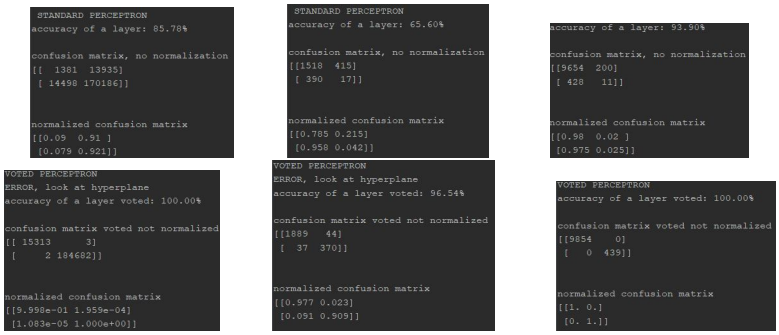

Figure 2: immagini relative ai grafici di differenza di accuratezza, dove la funzione del Perceptron

6 Conclusione

I risultati da noi studiati portano alla conclusione che ci aspettavamo. Infatti per ogni dataset studiato e per ogni epoch in cui facciamo il training del Perceptron possiamo vedere che l'accuratezza del Voted Perceptron è mantenuta costante al 100 % ,mentre quella del Perceptron standard si ritrova ad essere, in generale, minore rispetto a quella del Voted, ma anche ad avere dei peggioramenti a seconda delle epoche di studio. Le oscillazioni del Perceptron sono dovute all'indice di linearità del dataset che stiamo studiando.