UNIVERSITÀ
DEGLI STUDI
FIRENZE

Relazione Elaborato Corso Intelligenza Artificiale

Implementazione di Perceptron Votato

Giuseppe Parrotta
giuseppe.parrotta@stud.unifi.it

Dipartimento di Ingegneria dell'Informazione,
Laurea in Ingegneria Informatica

February 6, 2022

Contents

1 Introduzione

E' stato richiesto di implementare l'algoritmo **Perceptron Votato**, creato da Y. Freund e R. Shapire (1999), e quindi di testarlo usando tre diversi dataset per la verifica del corretto apprendimento. L'80% di ogni dataset è stato impiegato per l'apprendimento, e la restante parte per il test, con il quale si verifica che il Perceptron riesca a fare il minor numero possibile di errori.

Tramite il Perceptron Votato, si possono risolvere problemi di categorizzazione, sia nel caso in cui il dominio degli attributi fosse **discreto** (decidendo quindi quali fossero quegli attributi definiti *positivi*, e quali *negativi*), sia nel caso in cui fosse **continuo** (in questo caso selezionando il valore oltre il quale si considerava *positivo*, oppure **negativo** se inferiore). Il codice per testare questo algoritmo è allegato come file *.pdf*. L'intero codice sorgente è scritto in Python, e sono state utilizzate alcune librerie quali *numpy* e *pandas*, oltre a *matplotlib* per stampare i grafici.

2 Perceptron votato

Viene qui mostrato lo pseudocodice del perceptron votato.

Training

Input: a labeled training set $\langle(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\rangle$
number of epochs T

Output: a list of weighted perceptrons $\langle(\mathbf{v}_1, c_1), \ldots, (\mathbf{v}_k, c_k)\rangle$

- Initialize: $k := 0$, $\mathbf{v}_1 := \mathbf{0}$, $c_1 := 0$.
- Repeat T times:
  - For $i = 1, \ldots, m$:
    - Compute prediction: $\hat{y} := \text{sign}(\mathbf{v}_k \cdot \mathbf{x}_i)$
    - If $\hat{y} = y$ then $c_k := c_k + 1$.
      else $\mathbf{v}_{k+1} := \mathbf{v}_k + y_i \mathbf{x}_i$;
      $c_{k+1} := 1$;
      $k := k + 1$.

Prediction

Given: the list of weighted perceptrons: $\langle(\mathbf{v}_1, c_1), \ldots, (\mathbf{v}_k, c_k)\rangle$
an unlabeled instance: $\mathbf{x}$

compute a predicted label $\hat{y}$ as follows:

$$s = \sum_{i=1}^{k} c_i \, \text{sign}(\mathbf{v}_i \cdot \mathbf{x}); \quad \hat{y} = \text{sign}(s).$$

Figure 1: Pseudocodice

L'implementazione mappa all'interno di un file *.json* gli attributi, e una volta creati, verranno direttamente ricaricati: questo per consentirne la loro eventuale modifica (ad esempio, cambiare il livello dopo il quale un attributo viene considerato positivo).

3 Dataset: Abalone

Ho usato qui un dataset che raccoglie alcuni attributi degli abaloni (gasteropodi di mare) e in base ad alcuni parametri dovrebbe essere in grado di dire se l'età è superiore all'età media di tutti gli abaloni nel dataset di train (che è circa 10 anni). In realtà la vera età è quella nel parametro "Rings" (questo di cui stiamo facendo la classificazione binaria) + 1.5. In questo dataset, il perceptron non sembra cavarsela bene, in quanto non riesce a trovare un iperpiano che separi quelli che hanno "Rings" maggiorni di 10 (circa)

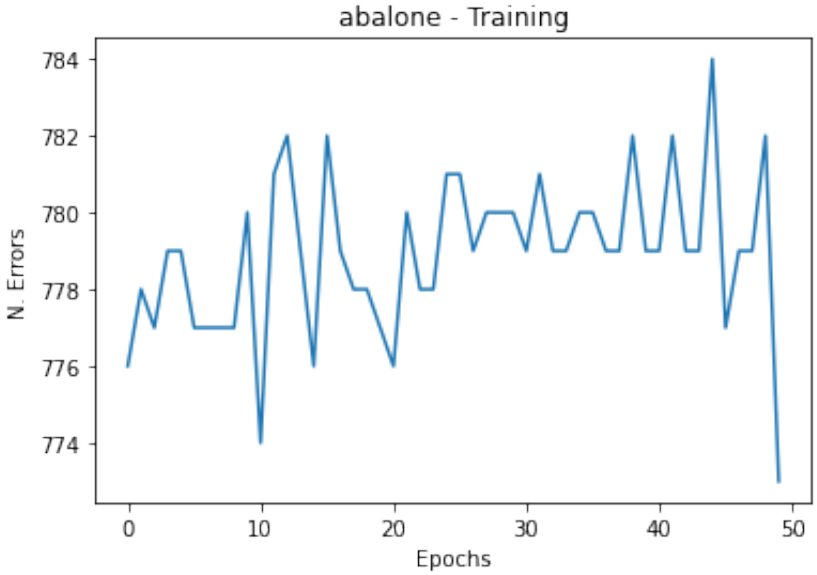

Figure 2: Abalone

Al crescere del numero di epoche, il numero di errori che il perceptron commette resta sempre circa lo stesso. Il dataset in questione non è linearmente separabile, e quindi questa forma di perceptron *antica* non è in grado di risolverlo.

	mistakes	correct	total
AVG	221	676	897
Vote	227	670	897

Figure 3: Abalone - Risultati test

Anche su questo piccolo dataset di test, si può vedere che il numero di errori non è trascurabile.

4 Dataset: Funghi

Questo secondo dataset ha domini discreti per ogni attributo. In questo caso, ho pensato di categorizzare gli attrbuti, e a ogni attributo di categoria *positivo* assegno il valore +1, e ad ognuno di quelli in categoria *negativo* il valore -1. In particolare, ho voluto allenare l'algoritmo a fargli capire se il fungo in questione fosse o meno velenoso.

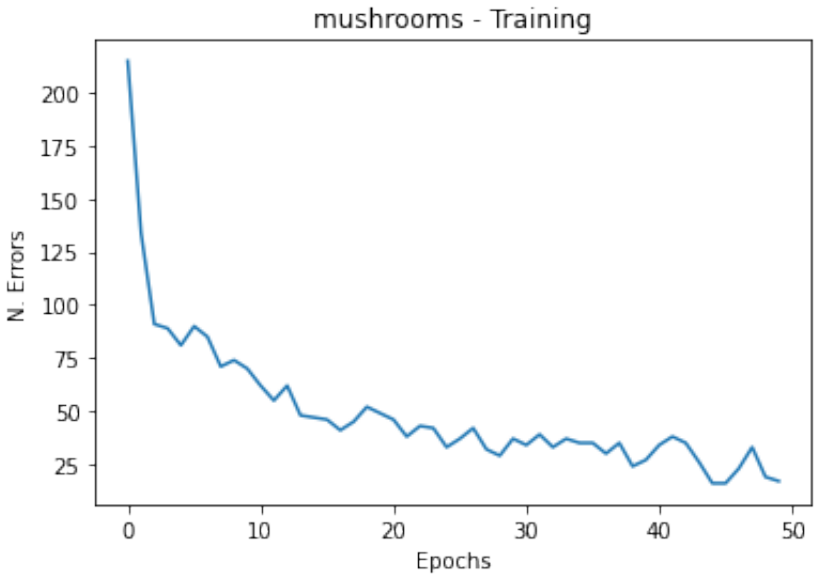

Figure 4: Funghi

Questo secondo perceptron sembra sbagliare molto meno se paragonato all'abalone.

	mistakes	correct	total
AVG	41	1603	1644
Vote	58	1586	1644

Figure 5: Funghi - Risultati test

In fase di test, andando a contare gli errori, si vede subito che sono molto meno rispetto sempre all'abalone.

5 Dataset: firewall

Questo ultimo dataset, con dominio degli attributi nei numeri Reali, presenta gli stessi risultati del dataset sugli abalone. Questo ci fornisce l'evidenza che usare il perceptron in modo diretto sui dati di questi dataset, non porta a risultati positivi.

6 Conclusioni

L'unico dataset che sembra dare risultati positivi è quello dei funghi, in quanto ha dominio degli attributi in un insieme discreto di valori. Selezionando quali valori considerare positivi e quali negativi, il perceptron implementato è infatti in grado di apprendere. Ergo per risolvere il problema dell'apprendimento su uno dei dataset in cui non ha funzionato (quindi quelli a valori nei Reali), si potrebbe tentare di fare *Bucketing* sul dataset, in modo di creare degli intervalli di valori. Questi intervalli potrebbero essere considerati positivi oppure negativi, come avviene nel dataset a valori discreti. L'approccio precedentemente seguito, prima del bucketing, è stato quello di definire un unico intervallo, e quindi classificare come positivi i valori uguali o al di sopra di quella soglia, e negativi quelli inferiori. Approccio evidentemente sbagliato.