Perceptron
vs.
Voted Perceptron

Tommaso Billi

5 aprile 2019

1 Introduzione

Nell'ambito del *machine learning*, il **Perceptron** è un algoritmo di supervised learning per classificatori binari. Un classificatore binario è in grado di decidere su un input, rappresentato attraverso un vettore di numeri, appartiene a una determinata classe.
L'algoritmo fu inventato nel 1957 da Frank Rosenblatt, e fin da subito riscosse un grande successo, promuovendo la ricerca nel settore. Nel 1969, con "Perceptrons", Marvin Minsky e Seymour Papert esposero i limiti di tale algoritmo, in riferimento alla capacità del Perceptron di riuscire ad identificare solamente dati di classi linearmente separabili. Minsky e Papert dimostrarono inoltre l'impossibilità per questo algoritmo di imparare da una funzione XOR.

In questa relazione si confronteranno le prestazioni degli algoritmi **Perceptron** e **Voted Perceptron**, applicando i due algoritmi a diversi dataset, e riferendoci in particolare ad accuratezza e matrice di confusione.

2 Perceptron

L'algoritmo permette di identificare la classe di appartenenza di un input $\mathbf{X}$ attraverso una funzione di previsione $f(\mathbf{X})$. La classe di appartenenza è rappresentato come una *label*, che assume valori $\in \{-1, 1\}$.

La fase di training, in cui all'algoritmo vengono passati dei valori di cui sappiamo a priori la classe di appartenenza, consiste nel calcolare il valore della funzione di previsione $f(\mathbf{X}) = \text{sign}(\mathbf{w} \cdot \mathbf{x} + b)$, dove $\mathbf{w}$ è il vettore dei pesi, inizialmente inizializzato a zero, e b è il bias. Nel caso in cui il valore predetto risultasse uguale al valore della label corrispondente, non è necessario apportare alcuna modifica; in caso contrario,

i valori di $\mathbf{w}$ e b vengono aggiustati in modo da rendere più probabile il successo di una successiva previsione.
Nel 1962 Novikoff dimostrò che, in caso di dataset linearmente separabile, il Perceptron converge in un numero finito di passi.

Nella fase di testing viene "indovinata" la classe di appartenenza di un input $\mathbf{X}$ utilizzando la stessa funzione di predizione di cui fa uso l'algoritmo durante il training.

3 Voted Perceptron

Il *Voted Perceptron* riprende l'idea del Perceptron, ma in esso si mantiene in memoria una lista di tutti i vettori di previsione generati ad ogni errore. Per ognuno di questi vettori, un ulteriore vettore $\mathbf{c}$ ci dice quante volte di fila una predizione è stata corretta durante la fase di training. In tal modo, questo algoritmo è in grado di effettuare previsioni più accurate rispetto al Perceptron standard.
La fase di testing del Voted Perceptron si distacca notevolmente da quella del normale Perceptron, in quanto il valore della funzione di predizione $f(\mathbf{X})$ è dato dalla somma dei segni del prodotto del vettore dei pesi moltiplicato per l'input $\mathbf{X}$ e moltiplicato a sua volta per il corrispondente numero di previsioni corrette:

$$s = \sum_{i=1}^{k} c_i \cdot \text{sign}(\mathbf{v}_i \cdot \mathbf{X}); \quad f(\mathbf{X}) = \text{sign}(s) \tag{1}$$

4 Implementazione

Il codice è stato implementato in Python 2.7[1].

Il costruttore di DataSet crea un oggetto utilizzabile dai perceptron, a partire dal path del file del dataset. In particolare, per creare ad esempio un dataset utilizzabile a partire dal file "data_banknote_authentication.txt", sarà sufficiente chiamare `DataSet(filename="DataSets/data_banknote_authentication.txt")`.
Le classi Perceptron e Voted Perceptron sono implementate nei rispettivi file `.py`. Nel costruttore di tali classi si possono passare, oltre all'oggetto DataSet, i valori di bias e learning rate desiderati; in caso contrario, verranno utilizzati quelli di default.
Nel file `test.py` sono definiti i metodi `holdoutValidation()` e `kFoldCrossValidation()`. Nel primo caso è possibile definire la suddivisione tra training e test dataset, mentre per il secondo metodo si può scegliere il numero k di iterazioni da compiere, passandolo come primo argomento. Sempre in questo file è possibile modificare il numero di *epoch* da utilizzare nella fase di training dei due perceptron.
In questo esperimento si è utilizzata soltanto la k-fold cross validation, in quanto in questo modo si lavora su diverse suddivisioni del dataset, piuttosto che soltanto con una.

[1] Il codice è reperibile all'indirizzo: *https://github.com/tommasobilli/Perceptron_vs._Voted_Perceptron*.

5 Risultati

5.1 Banknote Authentication dataset

	Test set size	Perceptron		Voted Perceptron	
		Mistakes	Accuracy	Mistakes	Accuracy
1st it.	274	4	98.54%	3	98.91%
2nd it.	274	7	97.45%	5	98.18%
3rd it.	274	6	97.81%	1	99.64%
4th it.	274	2	99.27%	2	99.27%
5th it.	276	8	97.10%	4	98.55%
Average	274.4	5.4	98.03%	3	98.91%

5.2 Phishing Websites dataset

	Test set size	Perceptron		Voted Perceptron	
		Mistakes	Accuracy	Mistakes	Accuracy
1st it.	2211	286	87.06%	176	92.04%
2nd it.	2211	289	86.93%	166	92.49%
3rd it.	2211	252	88.60%	175	92.09%
4th it.	2211	212	90.41%	158	92.85%
5th it.	2211	222	89.96%	159	92.81%
Average	2211	252.2	88.59%	166.8	92.46%

5.3 HTRU 2 dataset

	Test set size	Perceptron		Voted Perceptron	
		Mistakes	Accuracy	Mistakes	Accuracy
1st it.	3579	100	97.21%	97	97.29%
2nd it.	3579	112	96.87%	187	94.78%
3rd it.	3579	114	96.81%	95	97.35%
4th it.	3579	141	96.06%	103	97.12%
5th it.	3582	724	79.79%	560	84.37%
Average	3576.6	238.2	93.35%	208.4	94.18%

6 Conclusioni

I risultati riscontrati nei test rispecchiano le aspettative teoriche: in ciascuno dei dataset utilizzati i risultati migliori sono stati ottenuti con l'utilizzo del Voted Perceptron, nonostante anche la versione non voted sia comunque in grado di produrre ottimi risultati. Nel test con il Phishing Websites Dataset la differenza di performance tra i due algoritmi risulta ancora più evidente
I risultati ottenuti nei diversi esperimenti dipendono dalle caratteristiche del dataset, dal numero di esempi forniti e dagli attributi, ma in particolar modo dalla caratteristica del dataset stesso di essere più o meno linearmente separabile.

7 References

Freund, Y., Schapire R. E. (1999). *Large Margin Classification Using the Perceptron Algorithm*, pp. 277-295.

Norvig, P., Russell, S. J., *Artificial Intellingence: A Modern Approach. Third Edition.*, pp. 693-748, Prentice Hall, 2009.

Novikoff, A. B. (1962). On convergence proofs on perceptrons. *Symposium on the Mathematical Theory of Automata, 12*, 615-622. Polytechnic Institute of Brooklyn.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review, 65*, 386–407. (Reprinted in *Neurocomputing*, MIT Press, 1988.)

https://archive.ics.uci.edu/ml/datasets/banknote+authentication

https://archive.ics.uci.edu/ml/datasets/HTRU2

https://archive.ics.uci.edu/ml/datasets/Phishing+Websites

https://en.wikipedia.org/wiki/Perceptron