Progetto IA: Perceptron vs VotedPerceptron

Enrico Collini

April 6, 2018

1 Introduzione

Il Perceptron é uno dei primi algoritmi di supervised learning per classificatori binari ad essere stato implementato.
Nell'inizi degli anni '60 subito dopo il debutto, il Perceptron, il cui ideatore Frank Rosenblatt, produsse un enorme entusiasmo incoraggiando lo sviluppo della ricerca nell'ambito delle reti neurali.
Nel 1969 Marvin Minsky e Seymour Papert nel *Perceptrons* esposero i limiti di questo algoritmo in riferimento all'applicabilitá a Dataset non linearmente separabili e l'impossibilitá di produrre la funzione XOR.
In questo progetto andremo a comparare la versione originale con una modificata, il VotedPerceptron, applicandoli a vari Dataset. Per effettuare la nostra analisi sono stati scelti 3 dataset dall' UCI Machine Learning Repository. Sono stati scelti il banknote authentication Data Set, l'HTRU2 Data Set e il Diabetic Retinopathy Debrecen Data Set (I link sono forniti dell'apposita sezione per le References. Questo progetto é stato implementato in Python 3 e i files sono disponibili a https://github.com/EnricoCollini/PerceptronIA

2 Perceptron

L'algoritmo consiste nell'identificare la classe di appartenenza $\{-1, 1\}$ di un input X attraverso una funzione di previsione $f(x)$.
Per implementare l'algoritmo sono stati fissati un Learning Rate, un numero massimo di iterazioni e il vettore dei pesi inizializzandolo con valori random.

La fase di training, in cui all'algoritmo vengono passati dei valori di cui conosciamo le lables di appartenenza, consiste nel calcolare la funzione di previsione $f(x) = sign(< w, x > +b)$ e controllare se il valore predetto é corretto. Se questo non si verifica, i pesi vengono aggiornati tenendo conto dell'input, della lable corretta e del learning Rate stabilito.
L'implementazione di questo é fornita nella funzione **train(self, dataset, lables, numIt, learningRate)**.

La fase di testing consiste nel calcolare il valore predicted mediante la funzione

di previsione $f(x)$ e l'implementazione corrispondente é fornita nella funzione **predict(self, w, b, x)**.

3 VotedPerceptron

Il VotedPerceptron apporta dei miglioramenti rispetto alla versione originale perché tiene conto di quante volte una data assegnazione dei pesi riesce a non sbagliare. Proprio a questa sua caratteristica deve il suo nome.
Per implementare quanto appena detto il codice del Perceptron é stato modificato assegnandogli 3 ulteriori liste: perceptrons, times, bias.

La fase di training é strutturata nella stessa maniera della versione originale, l'unica differenza é che di volta in volta l'algoritmo salva nelle liste i valori di interesse. L'implementazione corrispondente é fornita dalla funzione **train(self, dataset, lables, numIt, learningRate)**

Nella fase di Testing la funzione di previsione subisce delle notevoli modifiche. In particolare il valore di $f(x)$ calcolato sui pesi e i bias reperiti dalle nuove liste viene a sua volta moltiplicato per il tempo t corrispondente (reperito attraverso la lista times) iterazione per iterazione.
Successivamente viene calcolato il sign per decidere la classse di appartenenza. L'implementazione corrispondente é fornita dalla funzione **predict(self, b, x)**

Nell'implementazione del VotedPerceptron é stato fornito anche un metodo **set()** che reinizializza le liste perceptrons bias e times.

4 Datasets

Per ogni Dataset sono stati creati 3 file python:

- Dataset: per estrarre i dati dal corrispondente dataset
- DatasetTesting: per ottenere i risultati del testing
- DatasetPlotResult: per visualizzare i risultati

4.1 Dataset

In questo file é stata implementata una strategia per salvare i valori del Dataset in un array numpy per poterli utilizzare al meglio nella successiva fase di testing. Il dataset contiene contemporaneamente gli input per i vari attributi e le lable di appartenenza. Sono stati forniti due metodi per separare dati e etichette:
createInputs(dataset) createLables(dataset)
Per effettuare la k-cross-fold-convalidation sono stati creati 4 gruppi e inizializzati con gli opportuni valori.

Sono stati forniti i metodi per accedere all'array numpy del dataset originle **get[nomeSpecificoDataset]Array** ,una versione shuffled **get[nomeSpecificoDataset]ArrayShuffled** e i vari partizionamento.

4.2 DatasetTesting

Mediante la K-Fold-Cross-Convalidation sui 4 gruppi di partizionamento descritti nella sezione precedente é stato effettuato il testing mediante sia il Perceptron che il VotedPerceptron.
Inoltre é stata fornita un implementazione mediante la creazione di un insieme di indici utile per rapppresentare la matrice di confusione e l'accuratezza.
I metodi per ottenere i risultati sono **getResults()** e **getResultsVoted()**

4.3 DatasetPlotResult

Attraverso l'uso di matplot é stato realizzato un metodo per creare la matrice di confusione e una tabella contenente l'accuratezza in base al dataset e all'algoritmo implementato prendendo i risultati calcolati nella sezione precedente. Il metodo implementato é **plotResults()**

5 TestingAll

Questo file python organizza l'esecuzione dei metodi plotResult() che chiamano i vari metodi di testing in base ai dataset e all'algoritmo scelto.

6 Risultati Sperimentali

BankPerceptron	Classe1	Classe2	Totale
Classe 1	720	42	762
Classe2	8	602	610
totale	728	644	1372
accuratezza			96%

BankVotedPerceptron	Classe1	Classe2	Totale
Classe 1	752	10	762
Classe2	3	607	610
totale	755	617	1372
accuratezza			99%

DiabeticPerceptron	Classe1	Classe2	Totale
Classe 1	254	286	540
Classe2	127	484	611
totale	381	770	1152
accuratezza			64%

DiabeticVotedPerceptron	Classe1	Classe2	Totale
Classe 1	366	174	540
Classe2	152	459	611
totale	518	633	1152
accuratezza			71%

HTRUPerceptron	Classe1	Classe2	Totale
Classe 1	15782	475	16257
Classe2	295	1346	1641
totale	16077	1821	17898
accuratezza			95%

HTRUVotedPerceptron	Classe1	Classe2	Totale
Classe 1	16071	186	16257
Classe2	283	1358	1641
totale	16354	1544	17898
accuratezza			97%

7 Analisi dei Risultati

I risultati ottenuti rispecchiano le aspettative teoriche: per ogni dataset abbiamo ottenuto dei risultati migliori utilizzando la versione voted del Perceptron, in particolare nel Dataset Diabetic Retinopathy Debrecen.
I diversi valori ottenuti dipendono dalle caratteristiche dei dataset, dal loro numero di attributi e dal numero di esempi, ma soprattutto dalla loro caratteristica di essere piú o meno linearmente separabili.

8 References

Large Margin Classification Using the Perceptron Algorithm 1999 Kluwer Academic Publishers. Manufactured in The Netherlands.
Russel Norvig "Artificial Intelligece A Modern Approach - Pearson
$https://en.wikipedia.org/wiki/Perceptron$
$https://archive.ics.uci.edu/ml/datasets/Diabetic+Retinopathy+Debrecen+Data+Set$
$https://archive.ics.uci.edu/ml/datasets/HTRU2$
$https://archive.ics.uci.edu/ml/datasets/banknote+authentication$
StackOverflow per i problemi sorti in fase di implementazione, in particolare nella fase per importare i dataset in python.
$https://matplotlib.org/api/_as_gen/matplotlib.pyplot.table.html$