Dual Perceptron

Lorenzo Ricci

Università degli studi di Firenze
Corso di Laurea in Ingegneria Informatica
Intelligenza Artificiale
A.S. 22/23

Sommario

1 Introduzione algoritmo

Il primo algoritmo iterativo per l'apprendimento di classificazioni lineari è la procedura proposta da Frank Rosenblatt nel 1956 per il Perceptron. E' una procedura *on-line* e *mistake-driven* che inizia con un vettore peso iniziale **w0** (solitamente inizializzato tutto a 0, **w0=0**) e si adatta ogni volta che un punto, che sta venendo addestrato, viene malclassificato dai pesi attuali. L'algoritmo aggiorna il vettore peso e il bias direttamente. Inoltre, questa procedura ha garantita la convergenza dall'esistenza di un iperpiano che classifica correttamente i punti su cui lo stiamo facendo addestrare, e in questo caso si dice che i dati sono *linearmente separabili*. Quindi, viceversa, se non esiste un iperpiano i dati si dicono non separabili. Si definisce *margine funzionale di un esempio* ($\mathbf{x}_i$,y_i) con rispetto all'iperpiano ($\mathbf{w}$,b), la quantità:

$$\gamma_i = y_i(\langle x_i, x_j\rangle + b)$$

e si nota che se $\gamma > 0$, implica una corretta classificazione di ($\mathbf{x}_i$,y_i). L'algoritmo Perceptron lavora quindi aggiungendo esempi di addestramento positivi classificati in modo errato o sottraendo quelli negarivi classificati in modo errato ad un vettore peso scelto all'inizio, in modo aribitrario. Senza perdita di generalità, se si assume che il vettore peso iniziale è un vettore zero, e che l'ipotesi finale sarà quella di essere una combinazione lineare dei punti di addestramento, possiamo ridefinire il vettore peso:

$$\mathbf{w} = \sum_{i=1}^{l} \alpha_i y_i \mathbf{x}_i$$

dove, dal momento in cui il segno del coefficiente di $\mathbf{x}_i$ è dato dalla classificazione di y_i, gli α_i sono volori positivi proporzionali al numero di volte in cui una classificazione errata di $\mathbf{x}_i$ ha causato l'aggiornamento del peso. Punti che hanno causato pochi errori avranno un valore più piccolo di α_i, viceversa punti più difficili avranno questo valore più grande. Quindi, fissato un set di addestramento S, si può pensare al vettore α come rappresentazione alternativa dell'ipotesi in coordinate diverse o duali.[1]

2 Implementazione

Lo scopo del seguente progetto è quello di implementare l'algoritmo nella sua forma duale, permettendo l'uso di funzioni kernel al posto del prodotto scalare ⟨xi, xj⟩.

Given training set S
$\boldsymbol{\alpha} \leftarrow \mathbf{0}$; $b \leftarrow 0$
$R \leftarrow \max_{1 \le i \le \ell} \|\mathbf{x}_i\|$
repeat
  for $i = 1$ to ℓ
    if $y_i \left(\sum_{j=1}^{\ell} \alpha_j y_j \langle \mathbf{x}_j \cdot \mathbf{x}_i \rangle + b \right) \le 0$ then
      $\alpha_i \leftarrow \alpha_i + 1$
      $b \leftarrow b + y_i R^2$
    end if
  end for
until no mistakes made within the *for* loop
return $(\boldsymbol{\alpha}, b)$ to define function $h(\mathbf{x})$ of equation (2.1)

Table 2.2: **The Perceptron Algorithm (dual form)**

[1] Cristianini, N.,& ShaweTaylor,J.(2000). An introductionto support vector machines and other kernelbased learning methods. Cambridge university press.

2.1 KernelFunctions

La classe KernelFunctions fornisce tre metodi, uno per ogni tipologia di funzione kernel.

- Linear kernel: prende in ingresso una matrice X e ritorna il prodotto scalare tra X e X trasposta.
- Polynomial kernel: prende in ingresso una matrice X, un paramentro c che viene sommato al prodotto scalare come nel kernel lineare e un ultimo paramento d che definisce il grado del polinomio.
- RBF kernel: prende in ingresso una matrice X e un parametro *gamma* e ritorna l'esponenziale della norma negativa moltiplicata per -*gamma*, il cui tutto è elevato al quadrato.

Per l'implementazione delle operazioni matematiche all'interno di questi metodi è stata utilizzata la libreria "numpy".

2.2 MyDualPerceptron

La mia implementazione dell'algoritmo consiste nella classe MyDualPerceptron. Il costruttore prende in ingresso la tipologia di kernel che si vuole utilizzare, ovvero un numero: 1 per il kernel lineare, 2 per il kernel polinomiale e 3 per l'rbf kernel. Lo pseudocodice mostrato in Table 2.2 viene implementato nel metodo *train()*, che prende in ingresso X, y, epochs. In particolare:

- X: Porzione di data set, ovvero i dati che verrano utilizzati per addestrare il vettore alpha e il bias.
- y: Porzione di data set, ovvero il vettore dei target di addestramento.
- epochs: il numero di epoche(iterazioni) del ciclo *for* sul quale si vuole addestrare l'algoritmo.

Inoltre all'interno del metodo *train()*, in base al tipo di kernel scelto dall'utente vengono mappati i dati di addestramento X e quindi creata la matrice K (nsamples,nsamples). Infine, si utilizza un ciclo *for* per implementare il repeat dal quale si uscirà quando all'interno del ciclo for interno non ci saranno errori. Una volta eseguita la funzione di *train()* che addestra i propri parametri alpha e b (bias), verrà chiamato il metodo *predict()* che prende anch'esso in ingresso X e y, i quali però non saranno uguali a quelli del *train()*, ma saranno un'altra porzione di data set che viene utilizzata per testare l'accuracy dell'algoritmo nel predire i dati del target in output.

2.3 Main

Lo scopo principale della classe *Main* è quello di leggere uno dei quattro data set messi a disposizione. Anche questi potranno essere scelti scrivendo un numero a scelta, in particolare:

- 1: Breast Cancer Wisconsin (Diagnostic)
- 2: Adult
- 3: Heart Disease
- 4: Rice (Cammeo and Osmancik)

Da ogni data set vengono prese due porzioni: la prima X, che contiene tutte le colonne e le righe dei dati che servono all'algoritmo per dedurre il problema di classificazione binaria, mentre la seconda y che contiene il vettore dei target. Il programma sfrutta: la libreria "numpy" per eseguire l'operazione *where()*, che modifica i valori interni in base a un vincolo per impostarli ad 1 o -1, la libreria "pandas" per leggere i data set in formato "csv" e prendere le porzioni di data set interessato grazie al metodo *iloc()* e infine la libreria "sklearn" per i metodi *train_test_split()* e *accuracy_score()* per, rispettivamente, dividere X e y in un parte destinata all'addestramento e una parte invece destinata al test e al calcolo dell'accuracy della predizione finale. Successivamente vengono chiamati i metodi *train()* e *predict()* per l'esecuzione dell'addestramento dell'algoritmo e la predizione sui dati di test. Inoltre, viene calcolato anche il tempo che l'algoritmo impiega a svolgere i propri metodi, grazie ad un ulteriore libreria "timeit".

3 Analisi

All'interno di questa sezione viene analizzata l'accuracy dell'algoritmo DualPerceptron nel riuscire a predire i dati di test dei vari data set, utilizzando le varie tipologie di kernel. Di seguito viene mostrata una tabella con tutti i calcoli dell'accuracy. Analizzando nel dettaglio i risultati e supponendo che la mole di dati di addestramento sia il 75% e il 25% di test e che l'algoritmo è stato addestrato su 1000 epoche, vediamo che:

- Breast Cancer Winsconsin (Diasgnostic): Questo data set contiene 30 colonne(n_features) e 569 righe(n_samples) e ha lo scopo di utilizzare le 30 caratteristiche delle cellule presenti nei campioni per classificarli in due categorie: benigni o maligni. Sia il kernel lineare che polinomiale non hanno ottimi risultati, mentre il kernel rbf riesce ad ottenere un risultato ottimo con un valore di gamma>=17. Con valori di gamma più piccoli il vettore predetto *y_pred* ritornava in alcuni punti il valore 0.

- Adult: Questo è il più grande data set con 14 colonne e 32.581 righe, e ha lo scopo di predire, in base a dati di tipo demografico di persone adulte, se un individuo guadagna più o meno di 50.000 euro. In questo caso tutte le tipologie di kernel hanno fornito un perfetta accuracy. Da notare però, l'elevato tempo di addestramento e predizione dei dati.

- Heart Disease: Questo data set contiene 13 colonne e 1026 righe e ha lo scopo di classificare i pazienti in base alla presenza o assenza di malattie cardiache. Sia il kernel lineare che polinomiale non hanno dato buoni risultati, anche con valori di d alti l'accuracy non cambia. L'rbf kernel invece non ha fornito un risultato ottimo ma, con valori di gamma compresi tra 1/100 e 1/20, ma accettabile.

- Rice (Cammeo and Osmancik): L'ultimo data set contiene 7 colonne e 3810 righe e ha lo scopo di prevedere la composizione chimica del riso in base alle sue caratteristiche geografiche e varietali, in particolare se è "Cammeo" o "Osmancik". Notiamo in questo data set, come in Adult, una grande necessità di tempo nell'addestrare i dati. Migliori prestazioni da parte del kernel lineare rispetto a quello polinomiale ma comunque poco accettabili. Invece con valori di gamma molto alti l'rbf riesce a predire con una perfetta accuracy.

Dai risultati sperimentali si nota subito una migliore capacità generale da parte del kernel rbf di separare i dati, rispetto a quello lineare e polinomiale. I risultati non ottimali degli altri due potrebbero dipendere da un scarsa capacità di quest'ultimi nel separare dati di maggiore difficoltà, risultati migliori potrebbero essere ottenuti aumentando il numero di epoche con cui viene addestrato l'algoritmo. Questa conclusione nasce dall'analisi del vettore alpha nel momento in cui ha finito il suo addestramento e che possiede al suo interno molti valori pari a 1000. Questo risultato significa che le epoche non sono state sufficienti alla classificazione di quei punti, di conseguenza un aumento di epoche sarà necessario per migliori prestazioni future, ovviamente a discapito del tempo di esecuzione. Quest'ultimo è elevato per i metodi *train()* e *predict()* per i data set Adult e Rice può dipendere da due fattori: il primo è il maggiore numero di esempi(righe), in particolare Adult, e il secondo invece una maggiore difficoltà dei dati stessi.

	Breast Cancer Winsconsin	Adult
Linear Kernel		
Polynomial Kernel		
Radial Basis Function		

Table 1: Prima tabella

	Rice & Cammeo	King-Rook_vs_King-Pawn
Linear Kernel	Time take to train the data = 8178s Time taken to predict the data = 2.72s Accuracy sul set di test: 0.87	Time taken to train the data is 7245s Time taken to predict the target is 1.4s Accuracy sul set di test: 0.9687
Polynomial Kernel	(d=2):Time taken to train the data is 24s Time taken to predict the target is 2.45s Accuracy: 1.0	(d=2):Time taken to predict the data is 302s Time taken to predict the data is 2.1s Accuracy sul set di test 0.9671
Radial Basis Function		(gamma=1): Time to train: 91.4s Time to predict: 4.3s Accuracy sul set di test: 0.93

Table 2: Seconda tabella

Nel codice fornito, ho utilizzato il pacchetto geometry per impostare le dimensioni del foglio A4, e ho definito i margini a 25 mm su tutti i lati. In questo modo, le tabelle saranno adatte per essere inserite in una relazione. Assicurati di includere i pacchetti multirow e geometry nel tuo documento LaTeX per utilizzare il codice correttamente. Puoi anche personalizzare ulteriormente le dimensioni dei margini o altri aspetti delle tabelle in base alle tue esigenze.