



Dual Perceptron (dual form)

Lorenzo Ricci

Università degli studi di Firenze
Corso di Laurea in Ingegneria Informatica
Intelligenza Artificiale

Sommario

1	Introduzione algoritmo	3
2	Implementazione	3
2.1	KernelFunctions	4
2.2	MyDualPerceptron	4
2.3	Main	5
3	Analisi	5

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 $\mathbf{w}\mathbf{0}$ (solitamente inizializzato tutto a 0, $\mathbf{w}\mathbf{0}=\mathbf{0}$) 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 si sta 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 *marginale funzionale di un esempio* (\mathbf{x}_i, y_i) con rispetto all'iperpiano (\mathbf{w}, b) , la quantità:

$$\gamma_i = y_i(\langle \mathbf{x}_i, \mathbf{w} \rangle + b)$$

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 negativi classificati in modo errato ad un vettore peso scelto all'inizio in modo arbitrario. 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 valori 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.

2 Implementazione

Lo scopo del seguente progetto è quello di implementare l'algoritmo nella sua forma duale descritto in (Cristianini & Shawe-Taylor 1999), permettendo l'uso di funzioni kernel al posto del prodotto scalare $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$.

```

Given training set  $S$ 
 $\alpha \leftarrow \mathbf{0}; b \leftarrow 0$ 
 $R \leftarrow \max_{1 \leq i \leq \ell} \|\mathbf{x}_i\|$ 
repeat
  for  $i = 1$  to  $\ell$ 
    if  $y_i \left( \sum_{j=1}^{\ell} \alpha_j y_j \langle \mathbf{x}_j \cdot \mathbf{x}_i \rangle + b \right) \leq 0$  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  $(\alpha, b)$  to define function  $h(\mathbf{x})$  of equation (2.1)

```

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

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 parametro c , che viene sommato al prodotto scalare come nel kernel lineare e un ultimo parametro d che definisce il grado del polinomio.
- RBF kernel: prende in ingresso una matrice X e un parametro γ 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 verranno utilizzati per addestrare il vettore α 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 anche lui 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, supponendo che la mole di dati di addestramento sia il 75% e il 25% di test, vediamo che:

- Breast Cancer Wisconsin (Diagnostic): Questo data set contiene 30 colonne(n_features) e 569 righe, il cui scopo è quello di utilizzare le 30

	Linear_kernel	Polynomial_kernel	RBF_kernel
Breast Cancer Wisconsin (Diagnostic) con n_features = 30	37,8%	(c=1, d=2) : 43% (c=1, d=3) : 38% (c=1, d=n_features) : 35%	Con un valore di gamma >= 17: 100%
Adult con n_features = 14	100 % Time taken to train the data is 340s Time taken to predict the target is 18s	Risultati con (c=1, d=2): 100% Time taken to train the data is 414s Time taken to predict the target is 22s Risultati con (c=1, d=3): Lasciato andare per 1 ora ma non ha dato risultato	Con gamma=1: 100% Time taken to train the data 5840s and to predict 522s
Heart Disease con n_features=13	50%	Risultati con (c=1 e d=3): 51% Risultati con (c=1 e d=2): 52,5%	Riassegnando con 1/100 <gamma <1/20 risultati buoni tutti intorno all'85%
Rice (Cammee and Osmancik) con n_features=7	58% Time taken to train the data is 2371s Time taken to predict the target is 0.26s	Risultati con (c=1, d=3): 42% Time taken to train the data is 2333s Time taken to predict the target is 0.578s	Con gamma=1000: 100%

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 $\gamma \geq 17$.

- Adult: Questo è il più grande data set con 14 colonne e 32.581 righe, il cui scopo è quello di predire, in base a dati di tipo demografico di persone adulte, se un individuo guadagna più o meno di 50.000 euro.
- Heart Disease: Questo data set contiene 13 colonne e 1026 righe, il cui scopo è quello di classificare i pazienti in base alla presenza o assenza di malattie cardiache.
- Rice (Cammee and Osmancik): Questo data set contiene 7 colonne e 3810 righe, il cui scopo è prevedere la composizione chimica del riso in base alle sue caratteristiche geografiche e varietali, in particolare se è "Cammee" o "Osmancik".

TODO: Da scrivere le conclusioni sui dati e l'itemize sopra...