Progetto AI: Kernel Voted Perceptron

Riccardo Bonini

13 giugno 2020

1 Introduzione

Lo scopo del progetto è quello di ricreare un Voted Perceptron in forma duale, che sfrutti una funzione Kernel per evidenziare come il passaggio in dimensioni superiori comporti una separabilità lineare maggiore, e meno errori in fase di allenamento di conseguenza. Il dataset utilizzato è composto da elementi di 10 classi diverse, ma nell'esperimento verrà trasformato in un dataset di elementi di sole 2 classi. Poniamo l'evidenza su come l'utilizzo di una funzione Kernel permetta di aumentare la separabilità lineare del dataset, e portiamo un esempio su una coppia di classi. Verranno stampati i grafici del Test Error e del numero di errori durante la fase di training, per ciascuna dimensione.

Il codice del Voted Perceptron si attiene alla sua forma duale: il vettore dei pesi risulta pari a $\mathbf{v}_k = \sum_{j=1}^{k-1} y_{ij} x_{ij}$. Quando, nel training e nel testing, deve essere effettuata una previsione di un'etichetta di un elemento x, avremo che il prodotto tra il vettore dei pesi e l'elemento x in questione sarà : $\mathbf{v}_k \mathbf{x} = \sum_{j=1}^{k-1} y_{ij} K(x_{ij}, x)$, dove $K(x_{ij}, x)$ rappresenta la funzione Kernel, nel nostro caso la **Polynomial Expansion**: $(1 + \mathrm{xy})^{d}$.

2 Componenti del progetto

2.1 Kernel Voted Perceptron

Nel file sono presenti la definizione della funzione Kernel utilizzata e della classe Kernel Voted Perceptron. Quest'ultima espone i metodi di training ***train*** e di previsione ***vote***.

2.2 Dataset

Il dataset utilizzato per questo progetto è il dataset di Zalando chiamato **Fashion-MNIST**: è composto da 60000 elementi per la parte di training e 10000 per la parte di testing. Si tratta di un dataset ben più complesso del classico **MNIST**. Viene caricato tramite il file mnist-reader.py.

2.3 Main

All'interno del file Main.py viene eseguito l'esperimento. Prima di tutto viene caricato il dataset, che è fornito già diviso in parte di train e di test, con etichette e elementi separati. Di seguito, vengono scelte due classi tra le 10 disponibili, nel nostro caso 1 e 8, e vengono prelevati gli elementi delle classi scelte sia dal test set che dal training set, al fine di trasformare il dataset da multiclasse a binario. Il nuovo training set sarà quindi composto da 12000 elementi, ed il test set da 2000. Successivamente, creiamo 3 istanze di KernelVotedPerceptron, su 3 dimensioni diverse, e alleniamo ciascuna di esse per 10 epoche, tramite il metodo train. Durante il training, campioniamo per ogni decimo di epoca il valore di k, al fine di poter ricreare successivamente tutti i vettori dei pesi parziali: sfruttando la forma duale di quest'ultimo, difatti, è possibile ricreare il vettore dei pesi al campione temporale i-esimo semplicemente guardando in tale campione quanti errori (k), erano stati commessi. La funzione plot della libreria matplotlib permette di mostrare il grafico relativo al Test Error ed al numero di errori durante il training campionato ogni decimo di epoca. Passiamo poi alla previsione di etichette sul nostro test set, mediante il metodo vote: poichè il numero totale di campioni è pari a 99 (il campionamento inizia a partire da 1/10 di epoca), il metodo vote esprimerà per ogni istanza x un vettore di lunghezza 99, che comprende le previsioni per ogni vettore dei pesi parziale. L'accuracy del nostro voted perceptron viene calcolata in funzione dell'ultima delle 99 previsioni, in quanto rappresenta la votazione tramite il vettore dei pesi "finale".

3 Esecuzione e risultati

Come è immediatamente possibile notare dai grafici, maggiore è la dimensione del Voted Perceptron, minore è il numero totale di errori commessi, e la curva raggiunge il massimo più rapidamente. Questo perchè, in accordo con la teoria, lavorando in dimensioni maggiori il dataset è più linearmente separabile, di conseguenza gli errori commessi calano. Da notare inoltre come la differenza sia più marcata nel passaggio da dimensione 1 a dimensione 2, rispetto invece alla differenza tra dimensione 2 e 3.
Vediamo anche come la differenza tra dimensione 2 e 3 sia molto piccola, in quanto l'aumento di dimensione risulta avere effetti quasi nulli.
In generale il nostro Voted Perceptron mantiene un accuracy alta per tutte le dimensoni, indice che il suo training ed il suo guessing su nuovi elementi siano buoni: come notabile all'interno del metodo *vote*, ogni volta che dobbiamo prevedere l'etichetta di un nuovo elemento x, scriviamo il prodotto tra il vettore dei pesi v ed x stesso come un array, in questo modo trasformiamo il numero di calcoli da $O(k^2)$ in $O(k)$. Questo fa si che per computare le previsioni parziali per ogni campionamento di epoca non si debba ricalcolare ogni volta il vettore dei pesi.

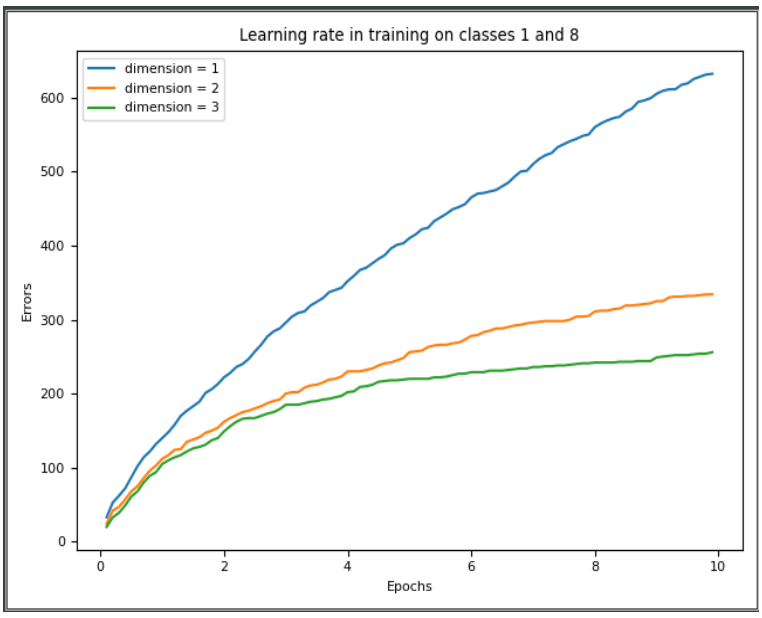

Figura 1: Learning rate

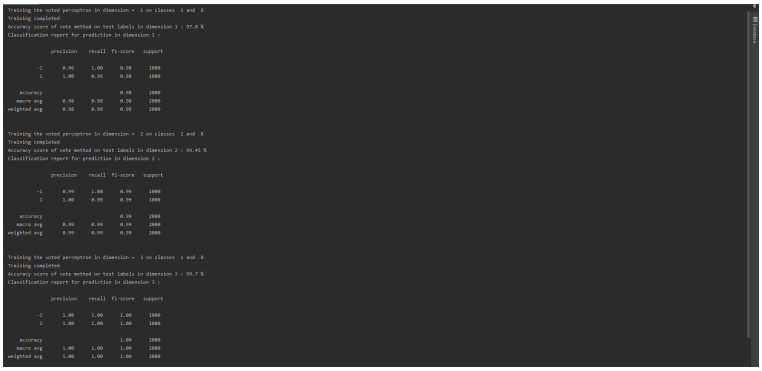

Figura 2: The output of the execution on classes 1 and 8

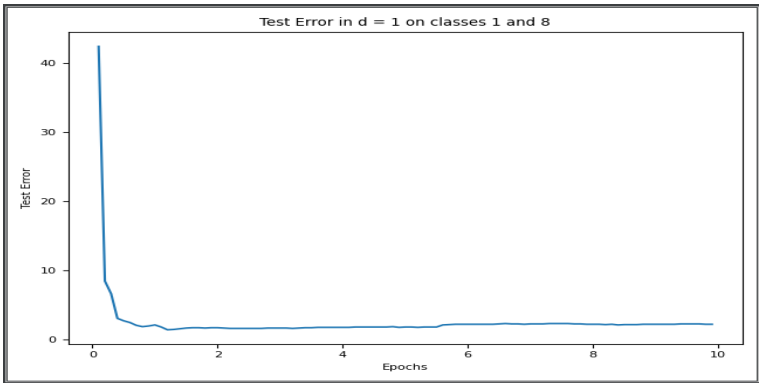

Figura 3: Test Error function in dimension 1

Figura 4: Test Error function in dimension 2

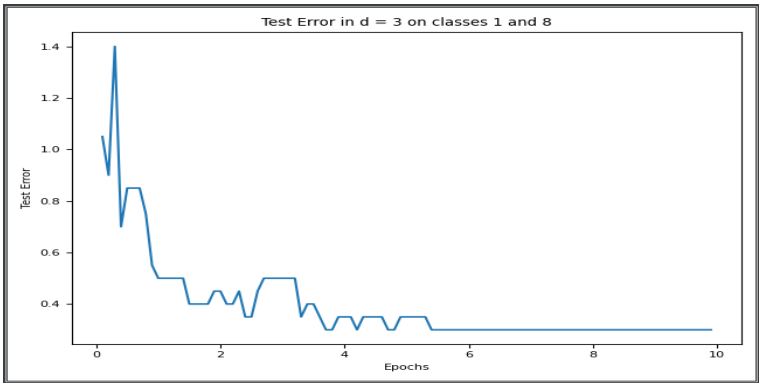

Figura 5: Test Error function in dimension 3