Progetto AI: Kernel Voted Perceptron

Riccardo Bonini

11 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 due esempi di allenamento su due coppie di classi differenti.

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**: $(1 + \text{xy})^{d}$.

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 0 e 5 prima, 1 e 9 successivamente per evidenziare la differenza, 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. Il training set viene infine permutato. Successivamente, creiamo 3 istanze di KernelVoted-Perceptron, su 3 dimensioni diverse, e alleniamo ciascuna di esse per 10 epoche, tramite il metodo train. La funzione plot della libreria matplotlib permette di mostrare il grafico relativo al numero di errori durante il training campionato ogni ventesimo di epoca. Passiamo poi alla previsione di etichette sul nostro test set, mediante il metodo vote, mostrandone l'accuracy.

3 Esecuzione e risultati

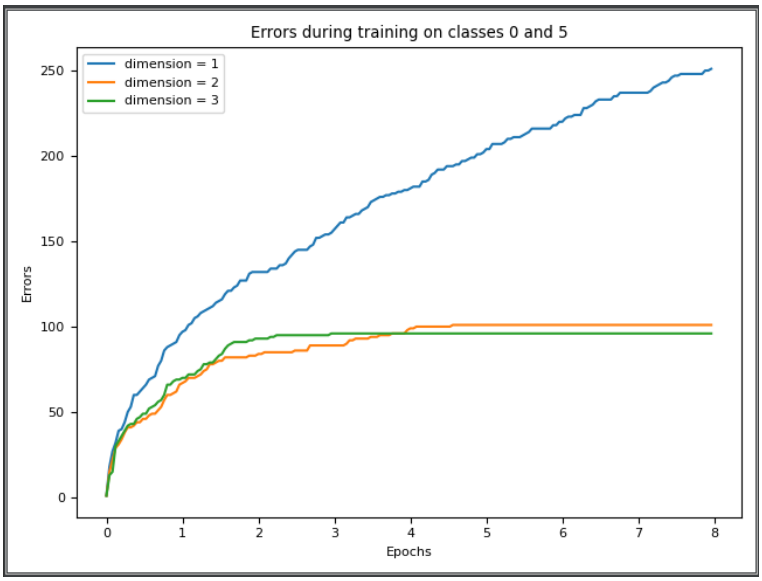

Figura 1: The output of the execution on classes 0 and 5

Come è immediatamente possibile notare dai grafici dei Test Errors, 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

Figura 2: The output of the execution on classes 0 and 5

= 2, rispetto invece alla differenza tra dimensione 2 e 3.
Altro aspetto chiave è il tempo di esecuzione: poichè nella forma duale il vettore dei pesi è rappresentato come sommatoria su tutti gli elementi sbagliati, maggiore il numero di errori, maggiore il tempo che è richiesto per effettuare la previsione delle etichette. Di conseguenza, la previsione risultante dall'allenamento su una dimensione superiore è nettamente più veloce rispetto ad una a dimensione inferiore. Come paragone, abbiamo allenato il Voted Perceptron anche sulle classi 1 e 9, un accoppiamento che risulta abbastanza linearmente separabile già in dimensione 1: si nota infatti dai grafici che il numero massimo di errori in assoluto è nettamente inferiore a quello dell'allenamento sulle classi 0 e 5, cosa che comporta un tempo di esecuzione di circa 2 minuti al fronte dei circa 15 minuti per quella precedente. 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 entrambi i casi, la differenza sostanziale è il passaggio da dimensione 1 a 2.
In generale il nostro Voted Perceptron mantiene un accuracy alta per tutte le dimensoni ed entrambe le casistiche, indice che il suo training ed il suo guessing su nuovi elementi siano buoni. In particolare, notiamo come l'accuracy del voto per le classi 1 e 9 sia praticamente identico nelle 3 dimensioni, poichè il passaggio di dimensione non viene "sentito" troppo, ovvero la separabilità lineare si mantiene circa uguale. Inoltre, comunque, vediamo che in entrambi i casi la differenza sostanziale è il passaggio da dimensione 1 a 2, in accordo con la teoria: dimensione 1 corrisponde a "nessuna espansione", il che comporta che la separabilità è quella di base, che può essere bassa come nel caso 0 e 5, oppure già alta come nel caso 1 e 9.

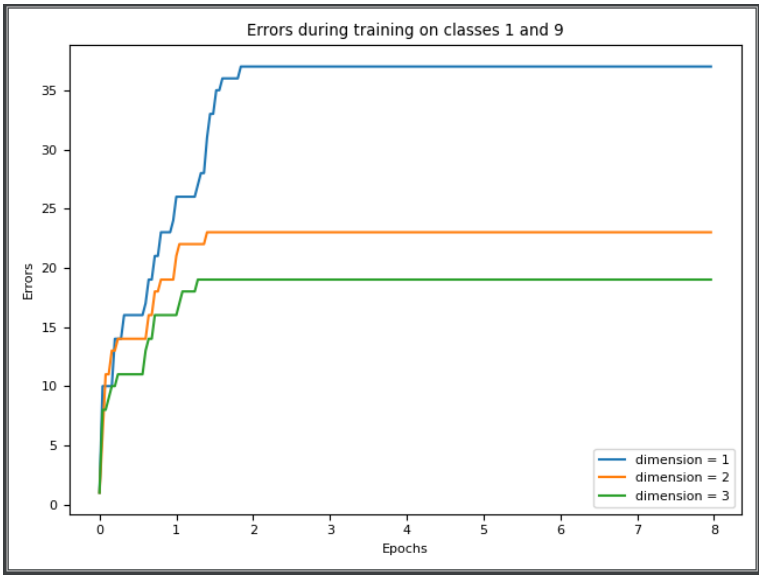

Figura 3: The output of the execution on classes 1 and 9

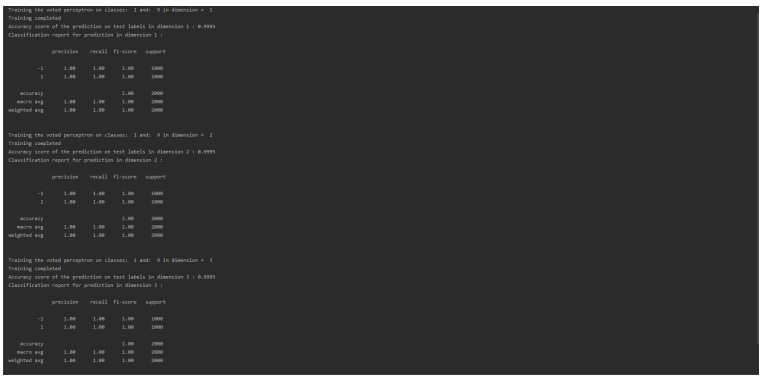

Figura 4: The output of the execution on classes 1 and 9