Classificare immagini di numeri civici

Giulia Di Costanzo

5 febbraio 2021

1 Introduzione

In questa relazione si descrive come classificare immagini di numeri civici utilizzando l'algoritmo Perceptron (della libreria scikit-learn), riportando l'errore di predizione sul training set e sul testing set, usando dimensioni del training set di 2^k, per k crescente da 10 fino al massimo valore compatibile con le risorse di calcolo disponibili.

2 Dataset SVHN

Nel dataset SVHN (nel formato cropped) sono presenti immagini di numeri civici delle case prese da Google Street View.
Il dataset contiene 10 classi, 1 per ogni digit. I digit da '1' a '9' hanno rispettivamente label da 1 a 9; il digit '0' ha label 10, perciò è stato convertito in 0. Sono presenti:

- 73257 digits per il training
- 26032 digits per il testing
- 531131 extra

3 Analisi

La funzione *get_file(file)* prende come input il file .mat da cui prende le immagini e restituisce:

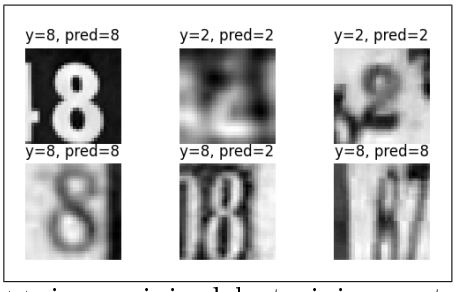
- **X**: una matrice 4D contenente le immagini
- **y**: un vettore contenente i label delle classi

3.1 Preprocessing

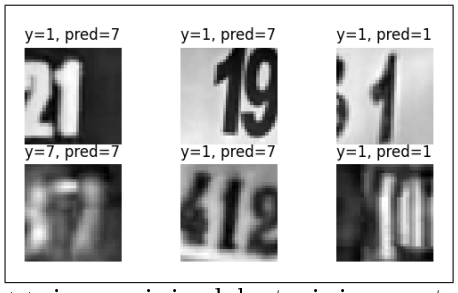
Per la fase di preprocessing delle immagini sono state implementate le funzioni *get_ images()*, *scale_ images()* e *balanced_ classes()*.
La funzione *get_ images(n_ img, file, digit1, digit2)* prende in input il numero di immagini, il file .mat e i due digits che si vogliono classificare.
In output restituisce due vettori:

- **x** che contiene le immagini in 1D
- **y** che contiene i label

Le immagini sono convertite prima in gray scale con la funzione *rgb2gray()* di scikit-image e poi con la funzione *flatten()* sono trasformate in un vettore a una dimensione.
La funzione *get_ train_ test_ sets()* crea il training set e il testing set (x_train, y_train, x_test, y_test). Le immagini del trainig set sono standardizzate con la funzione *scale_ images()*, usando *StandardScaler()* di scikit-learn, dopo aver bilanciato le classi con la funzione *balanced_ classes()*. Il bilanciamento delle classi è eseguito combinando i metodi di random under-sampling e random over-sampling, di imbalanced-learn, per non cambiare le dimensioni del training set (il numero di elementi della classe maggiore diminuisce mentre quello della classe minore aumenta).
Nelle immagini seguenti sono rappresentate le immagini del trainig e testing set dopo essere state trasformate in gray scale, riportando il valore della classe reale e quello predetto.



(a) immagini del training set, digits 2 e 8



(b) immagini del training set, digits 1 e 7

Figura 1: Es. immagini in gray scale

3.2 Classificazione

Il Perceptron è un algoritmo per l'apprendimento supervisionato di classificatori binari. Un classificatore binario è una funzione che può decidere se un input, rappresentato da un vettore di numeri, appartiene o meno a una classe specifica.
Dopo aver preprocessato le immagini, nella funzione *analysis(num, digit1, digit2)* è stato creato il classificatore Perceptron ed è stato addestrato usando il training set. Il modello istruito è stato usato per l'analisi del testing set: dopo aver calcolato l'accuratezza, sia per il training set che per il testing set, sono stati calcolati i relativi errori e restituiti dalla funzione *analysis()*.

4 Risultati

Nella funzione *main()* è stata fatta l'analisi usando dimensioni del training set crescente e dato che ogni sottoinsieme che crea il trainig set è un campione random, è stata eseguita una media su più trainig set della stessa dimensione. I risultati ottenuti sono riportati nelle tabelle sottostanti. Le dimensioni del training set sono di 2^k, per k crescente da 10 a 16; per il testing set la dimensione è il 30% di quella del training. I digits scelti sono 2 e 8 o 1 e 7. In queste tabelle sono state riportate le medie dell'errore su 5 iterazioni, per il training e il testing set.

Errore	
Training	Testing
32.754	47.941
40.469	50.513
43.921	49.963
44.431	48.034
45.282	49.310
45.190	48.298
45.084	47.237

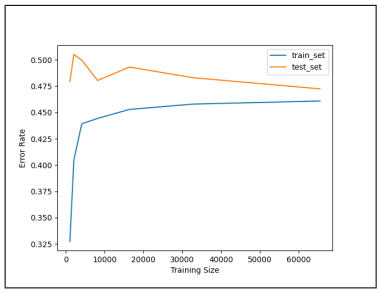


Figura 2: Digits 1 e 7

Errore	
Training	Testing
33.457	45.081
36.934	46.319
37.939	44.839
43.840	47.847
43.492	47.691
45.125	47.058
44.243	45.098

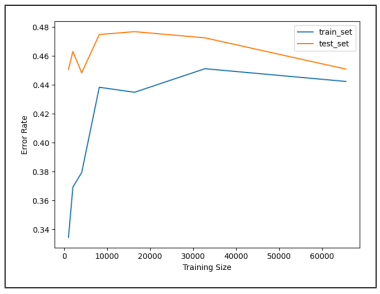


Figura 3: Digits 2 e 8

Nelle immagini seguenti sono stati riportati due esempi di matrice di confusione per due digits.

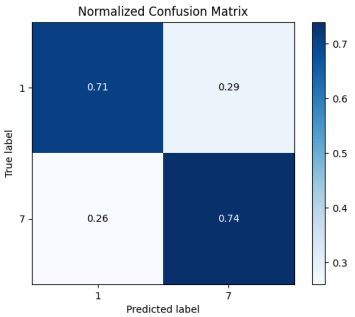


Figura 4: Matrice di confusione del training set, digits 1 e 7

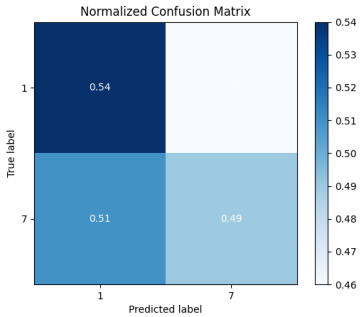


Figura 5: Matrice di confusione del testing set, digits 1 e 7

Come si può notare dalle tabelle sottostanti, i valori dell'errore variano per una stessa dimensione del training set, questo perchè le immagini sono prese casualmente in ogni iterazione. Nelle tabelle sono riportati due esempi di valori dell'errore sulle 5 iterazioni, rispettivamente per un training set di dimensione 1024 e di dimensione 4096.

Errore (dim=1024)	
Train	**Test**
39.253	42.0201
37.979	49.837
42.384	39.739
37.301	47.557
39.354	49.511

Errore (dim=4096)	
Train	**Test**
42.066	48.779
42.138	46.417
42.211	46.661
41.601	45.114
43.262	43.974

5 Conclusione

In conclusione usando l'algoritmo Perceptron per calcolare l'errore di predizione del training set e del testing set è stato notato un miglioramento del secondo, aumentando le dimensioni del training set. Inoltre come si può notare dai grafici, inizialmente l'errore nel training set è minore di quello del testing set ma con l'aumentare della dimensione del training, l'errore tende ad essere lo stesso.