Horjea Cosmin-Marian
Grupa: 243

Documentatie Proiect IA

Date de intrare

Datele sunt imagini cu dimensiunea de 32x32, alb negru, pe care le putem citi ca un vector bidimensional cu numere cuprinse între 0 și 255, care arată cat de mult alb se afla într-un pixel.

Prelucrare

Preluam imaginile și le transformăm dintr-un array bidimensional în unul unidimensional folosind functia **reshape(),** astfel fiecare imagine este un vector de 1024 de element, fiecare fiind un numar intre 0 si 255 și îl putem folosi pentru antrenarea modelelor din sklearn.

De asemenea folosim standardizarea standard pentru setul de antrenare și de validare

```
from sklearn import preprocessing

scaler = preprocessing.StandardScaler()
scaler.fit(X_train)

X_train,X_test=scaler.fit_transform(X_train),scaler.fit_transform(X_test)
```

Practic scădem imaginea medie din fiecare imagine din setul nostru de date și împărțim la deviația standard

Modele incercate

Prima data am folosit un clasificator **Naive Bayes**

```
from sklearn.naive_bayes import GaussianNB

clfGaussian = GaussianNB()
clfGaussian.fit(X_train, y_train)

clfGaussian.score(X_test,y_test)
```

Am obtinut un scor de **0.3992** pe setul de validare

Următoarea încercare a fost un clasificator care folosește metoda **KNN**

```
from sklearn.neighbors import KNeighborsClassifier

clfKNN = KNeighborsClassifier(n_neighbors=20)
clfKNN.fit(X_train, y_train)
clfKNN.score(X_test,y_test)
```

Horjea Cosmin-Marian
Grupa: 243

Am obtinut scorul de **0.4864** și după aceea am decis sa încerc sa modific parametrul de **n_neighbours**.

```
for k in [1,5,10,20,50]:
    clfKNN = KNeighborsClassifier(n_neighbors=k)
    clfKNN.fit(X_train, y_train)
    print('k=',k,' score=', clfKNN.score(X_test,y_test))
```

Cu care am obtinut
k= 1 score= 0.4416
k= 5 score= 0.4772
k= 10 score= 0.4906
k= 20 score= 0.4864
k= 50 score= 0.4716

Urmatorul model a fost un **RandomForestClassifier**:

```
clfRandomForest = RandomForestClassifier(max_depth=50, random_state=1)
clfRandomForest.fit(X_train, y_train)
clfRandomForest.score(X_test,y_test)
```

Care a obținut un scor de **0.6226** și am încercat sa variez parametrul de **max_depth**

```
for depth in [10,25,50,100]:
    clfForest = RandomForestClassifier(max_depth=depth, random_state=1)
    clfForest.fit(X_train, y_train)

    print('d= ',depth,' score= ',clfForest.score(X_test,y_test))
```

Iar rezultatele au fost
d= 10 score= 0.5916
d= 25 score= 0.6278
d= 50 score= 0.6266
d= 100 score= 0.6262

Urmatorul a fost un **SGDClassifier**:

```
from sklearn.linear_model import SGDClassifier
sgd_clf = SGDClassifier(random_state=1,verbose=10)
sgd_clf.fit(X_train, y_train)
sgd_clf.score(X_test,y_test)
```

Cu care am obtinut scorul **0.5294**
După aceea m-am gandit sa schimb parametrul de **loss**:

Horjea Cosmin-Marian
Grupa: 243

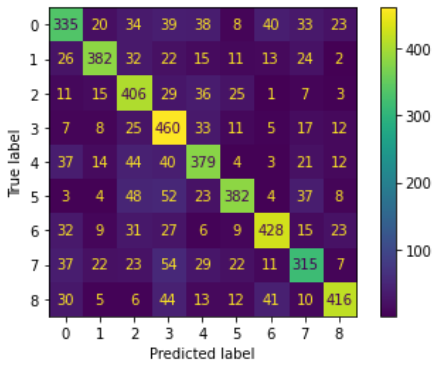
```
for lossFun in
['hinge','log','modified_huber','squared_hinge','perceptron']:
    sgd_clf = SGDClassifier(loss=lossFun,random_state=1)
    sgd_clf.fit(X_train, y_train)
    print('loss= ', lossFun, 'score= ', sgd_clf.score(X_test, y_test)
```

Iar rezultatele au fost
loss= hinge score= 0.5294
loss= log score = 0.4924
loss= modified_huber score= 0.4164
loss= squared_hinge score= 0.3978
loss= perceptron score= 0.527

După aceea am folosit un **MLPClassifier**:

```
from sklearn.neural_network import MLPClassifier
clfMLP=MLPClassifier(hidden_layer_sizes=(100, 100, 100,),
                     random_state=1, max_iter=100, verbose=10).fit(X_train,
y_train)
clfMLP.score(X_test, y_test)
```

Iar aici am obtinut un score de **0.7276**
Am folosit acest model pentru a face submit pe kaggle unde am obtinut **0.74080**
Matricea de confuzie arată așa:

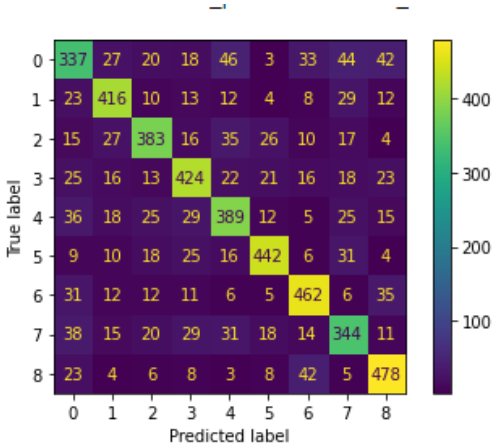


Următorul model a fost un **SVM**:

```
from sklearn.svm import SVC

clfSVM=SVC(C=1.0, verbose=True, kernel='rbf')
clfSVM.fit(X_train, y_train)
clfSVM.score(X_test, y_test)
```

Unde am primit un scor de **0.7378** pe setul de validare iar pe cel de testare de pe Kaggle **0.75040** iar matricea de confuzie arată astfel:



Horjea Cosmin-Marian
Grupa: 243

Tensorflow

Am decis sa folosesc Tensorflow cu Keras pentru a construi un model nou
O schimbare a fost aceea ca datele aveau acum dimensiunea de (32x32x1) pentru ca sunt imagini de 32x32 cu o singura culoare

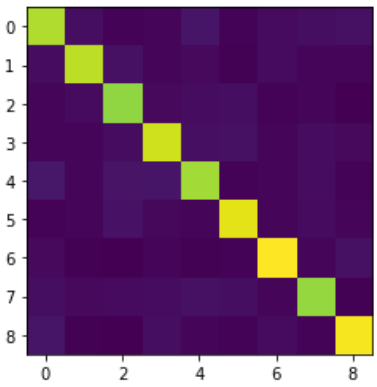
```
clfKeras=Sequential()
clfKeras.add(Conv2D(32, (3, 3), activation='relu', padding='same'))
clfKeras.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
clfKeras.add(keras.layers.Flatten())
clfKeras.add(Dense(100, activation='relu'))
clfKeras.add(Dense(9, activation='softmax'))

clfKeras.compile(loss='categorical_crossentropy',
              optimizer='adam', metrics=['accuracy'])

clfKeras.fit(X_train, y_train, epochs=20, validation_data=(
    X_test, y_test), batch_size=512)
```

Avem un model secvențial cu doua straturi convolutionare cu 32,respectiv 64 de filtre, mărimea kernel-ului de 3x3 și metoda de activare ‘relu’. Stratul “Flatten” care ne transforma datele în unu dimensionale. Acestea le pasam într-un layer cu 100 de unități iar în final într-un layer cu 9 unități deoarece încercăm sa prezicem clase de la 0 la 8.
Atunci cand compilam modelul, folosim ca funcție de loss ‘categorical_crossentropy’ care calculeaza cate de diferite sunt clasele prezise de model fata de clasele adevărate.
În final antrenam modelul timp de 20 de epoci cu un batch_size de 512

Pe setul de validare, modelul avea o acuratețe de **0.8288** și după ce am dat submit am obtinut **0.84560** pe clasamentul inițial iar după aceea un scor de **0.82613** pentru clasamentul final
Matricea de confuzie pentru acest model arată astfel:




```
array([[444,  19,   5,   8,  29,   4,  14,  19,  23],
       [ 17, 454,  22,   9,  12,   2,  15,   9,   6],
       [  6,  14, 421,  13,  17,  18,   5,   9,   0],
       [  7,   7,  17, 467,  21,  22,   6,  16,  12],
       [ 32,   9,  26,  28, 435,   4,   6,  17,   5],
       [  5,   6,  25,  11,   9, 484,   6,  15,   7],
       [ 13,   2,   1,   6,   2,   5, 504,   7,  23],
       [ 18,  13,  15,  17,  22,  18,   9, 424,   3],
       [ 28,   3,   1,  19,   7,   4,  15,   4, 498]])
```

Horjea Cosmin-Marian
Grupa: 243

Bibliografie

- https://scikit-learn.org/stable/modules/classes.html
- https://keras.io/api/
- https://www.tensorflow.org/api_docs/python/tf/keras/
- https://www.youtube.com/playlist?list=PLUl4u3cNGP63gFHB6xb-kVBiQHYe_4hSi