Coaje Florian-Petrut
Grupa 243

Documentatie

In cadrul competitiei am abordat urmatoarele modele : naïve-bayes, k-NN classifier, SVM si MLP. Imaginile sunt incarcate sub forma unei liste de 1024 de elemente (pixeli). Astfel, lista train_images este de forma 30000x1024, iar validation_images si test_images de 5000x1024.

1. Naïve-Bayes

Am impartit cei 256 de pixeli in 11 intervale, retinand capetele cu ajutorul functiei linspace din libraria numpy. Functia values_to_bins discretizeaza multimea de antrenare si de validare. Antrenam modelul pe datele de antrenare, apoi folosind functia score pe datele de validare obtinem o acuratete de 0.3916.

```
Scorul obtinut pe datele de validare este:
0.3916
Matricea de confuzie este:
[[ 63.  87.  22.  46.  89.  28.  43.  99.  93.]
 [ 14. 242.  23.  43.  15.  21.  24. 100.  45.]
 [ 21.  64. 148.  26.  44.  98.  31.  79.  22.]
 [ 19.  58.   8. 298.  34.  68.  23.  46.  24.]
 [ 20.  49.  40.  38. 180.  73.  20. 106.  28.]
 [  6.  15.  31.  54.  37. 337.  13.  52.  16.]
 [ 42.  64.  11.  48.  86.  29. 155.  70.  75.]
 [ 20.  53.  21.  33.  56.  22.  17. 283.  15.]
 [ 48.  54.  16.  48.  51.  44.  29.  35. 252.]]
```

2. K-NN Classifier

Am incercat implementarea unui 3-NN classifier care foloseste distanta L2. Modelul alege cei mai apropiati 3 vecini, numara aparitiile labelurilor lor si atribuie imaginii pe acela cu cele mai multe aparitii . Avand un set de date mare, modelul este foarte lent.

Am obtinut o acuratete de 0.22 pe setul de validare. Din aceste motive, am renuntat la a-l imbunatati.

3. SVM

Am normalizat datele folosind norma L2 . Modelul are ca parametri C = 2.0, deoarece alegand un C mai mare puteam ajunge la overfitting, kernel = ‘rbf’ si gamma default. Parametrul C sugereaza cat de mult este dispus modelul sa evite clasificarea gresita.

```
Scorul obtinut pe datele de validare este:
0.7416
Matricea de confuzie este:
[[340.  25.  27.  18.  35.   4.  46.  40.  35.]
 [ 19. 419.  11.  13.  11.   4.  13.  29.   8.]
 [ 14.  28. 389.  13.  27.  23.  15.  20.   4.]
 [ 15.  18.  18. 428.  19.  23.  15.  20.  22.]
 [ 35.  16.  30.  24. 388.  13.   2.  28.  18.]
 [  9.   7.  20.  26.  13. 449.   8.  22.   7.]
 [ 21.  20.  10.  11.   7.   6. 468.   8.  29.]
 [ 32.  15.  19.  23.  25.  19.  11. 366.  10.]
 [ 27.  11.   4.   7.   6.   7.  43.  11. 461.]]
```

De asemenea, am incercat si un SVM cu C= 1,5 , kernel = ‘rbf’ obtinand o acuratete mai mica . Odata cu reducerea parametrului C scade acuratetea, deoarece se ajunge la underfitting.

```
Scorul obtinut pe datele de validare este:
0.7362
Matricea de confuzie este:
[[336.  25.  27.  17.  38.   4.  49.  39.  35.]
 [ 17. 421.  12.  10.  12.   5.  13.  29.   8.]
 [ 13.  29. 388.  14.  27.  26.  14.  18.   4.]
 [ 17.  18.  16. 419.  21.  24.  16.  23.  24.]
 [ 36.  16.  30.  24. 385.  14.   2.  28.  19.]
 [  7.   8.  20.  29.  13. 447.   8.  22.   7.]
 [ 20.  19.  10.  11.   8.   6. 469.   7.  30.]
 [ 33.  17.  21.  28.  25.  18.  12. 355.  11.]
 [ 24.  11.   5.   8.   6.   6.  45.  11. 461.]]
```

Coaje Florian-Petrut
Grupa 243

4. MLP

Am aplicat o normalizare standard a datelor . Functia mlptrain primeste ca parametri :

- Datele de antrenare
- Etichetele datelor de antrenare
- Functia de activare
- Dimensiunea layerelor ascunse
- Rata de invatare
- Momentum-ul
- Numarul maxim de iteratii
- Parametrul alpha, pentru regularizare L2

Functia creaza un model MLP cu parametri dati si il antreneaza pe setul de date de antrenare. Rata de invatare este constanta.

Am incercat un model cu un numar mai mic de epoci sau hidden layers mai mic, de exemplu (100,100), dar am obtinut acuratete mai mica (<0.75). Pentru learning_rate_init de 0.001, antrenarea dura foarte mult (dupa o ora am renuntat).

Modelul care mi-a generat cea mai buna acuratete dintre incercarile mele are ca parametri: activation = ‘relu’, hidden_layer_sizes = (500, 500), momentum = 0.9, learning_rate_init = 0.01, max_iter = 2000, iar alpha este default. Am obtinut acuratetea de 0.7698, iar matricea de confuzie este urmatoarea:

Coaje Florian-Petrut
Grupa 243

```
0.7698
Matricea de confuzie este:
[[380.  14.  16.  15.  51.   7.  27.  31.  29.]
 [ 16. 437.  11.  11.   8.   6.   8.  25.   5.]
 [  8.  23. 399.  17.  35.  24.   3.  19.   5.]
 [ 13.  13.  13. 432.  26.  23.  14.  20.  24.]
 [ 33.  17.  26.  24. 408.   9.   4.  22.  11.]
 [  4.   6.  21.  22.  13. 452.   5.  33.   5.]
 [ 32.  10.  12.   9.   5.   8. 476.   5.  23.]
 [ 31.  17.  17.  27.  16.  22.   4. 383.   3.]
 [ 23.   3.   2.  10.   5.  10.  37.   5. 482.]]
```