

2^η ΕΡΓΑΣΙΑ ΓΙΑ ΤΟ ΜΑΘΗΜΑ
«ΝΕΥΡΩΝΙΚΑ ΔΙΚΤΥΑ-ΒΑΘΙΑ ΜΑΘΗΣΗ»

SUPPORT-VECTOR MACHINES

- Εφαρμογή στο dataset CIFAR-10
- Έλεγχος για διαφορετικές συναρτήσεις Kernell και παραμέτρους
- Σύγκριση απόδοσης με K-Nearest Neighbors



ΑΡΙΣΤΟΤΕΛΕΙΟ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΘΕΣΣΑΛΟΝΙΚΗΣ

Χρήστος Ιωαννίδης, τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

ΑΕΜ: 9397

ioannid@ece.auth.gr

Εισαγωγή

Σε αυτή την εργασία έγινε εφαρμογή του αλγορίθμου Support Vector Machine για την αναγνώριση/ταξινόμηση εικόνων από το dataset Cifar-10 της βιβλιοθήκης Keras σε 10 κλάσεις. (0: αεροπλάνο, 1: αυτοκίνητο, 2: πουλί, 3:γάτα, 4:τάρανδος, 5:σκύλος, 6: βάτραχος, 7:άλογο, 8:πλοίο, 9:φορτηγό

Ο κώδικας υλοποίησης του παραπάνω γράφτηκε σε Python και χρησιμοποιήθηκαν μέθοδοι από τις βιβλιοθήκες sklearn και keras. Τα dataset έγιναν load, στη συνέχεια reshape έτσι ώστε να έχουμε για τη κάθε εικόνα μονοδιάστατο πίνακα χαρακτηριστικών, και πιο συγκεκριμένα των στοιχείων του κάθε pixel, και πάνω σε αυτά έγινε η εκπαίδευση του μοντέλου, σε 50000 εικόνες-δείγματα πιο συγκεκριμένα.

Έπειτα, έγινε έλεγχος της απόδοσης της μεθόδου σε ένα set 10000 εικόνων-δειγμάτων για να αναλυθούν τα αποτελέσματα.

Η παραπάνω διαδικασία επαναλήφθηκε για διαφορετικές παραμέτρους των SVM, δηλαδή για τρεις διαφορετικές συναρτήσεις Kernel, για διαφορετικές τιμές γάμμα, και C.

Τέλος, εφαρμόστηκαν στα ίδια σετ οι αλγόριθμοι Πλησιέστερων Γειτόνων και Nearest Centroid.

Οι παράμετροι των SVM δοκιμάστηκαν σε μικρότερα σετ (train:5000, test: 1500) , και αφού καταλήξαμε στον καλύτερο δυνατό, εφαρμόστηκε και σε μεγαλύτερο σετ δεδομένων.

Έκθεση αποτελεσμάτων

Kernel: Linear, $\gamma=0,1$, $C=1$

```
accuracy: 0.298
precision: [0.27918782 0.38607595 0.24581006 0.20231214 0.23076923 0.12
0.36283186 0.3697479 0.42937853 0.38834951]
recall: [0.36912752 0.40131579 0.30136986 0.23178808 0.26277372 0.11904762
0.24404762 0.29530201 0.46060606 0.25477707]
      precision    recall  f1-score   support

     0         0.28         0.37         0.32         149
     1         0.39         0.40         0.39         152
     2         0.25         0.30         0.27         146
     3         0.20         0.23         0.22         151
     4         0.23         0.26         0.25         137
     5         0.12         0.12         0.12         126
     6         0.36         0.24         0.29         168
     7         0.37         0.30         0.33         149
     8         0.43         0.46         0.44         165
     9         0.39         0.25         0.31         157

 accuracy
macro avg         0.30         0.29         0.29         1500
weighted avg         0.31         0.30         0.30         1500

time: 100.1141575
```

$\gamma=1$

```
accuracy: 0.298
precision: [0.27918782 0.38607595 0.24581006 0.20231214 0.23076923 0.12
0.36283186 0.3697479 0.42937853 0.38834951]
recall: [0.36912752 0.40131579 0.30136986 0.23178808 0.26277372 0.11904762
0.24404762 0.29530201 0.46060606 0.25477707]
      precision    recall  f1-score   support

     0       0.28       0.37       0.32        149
     1       0.39       0.40       0.39        152
     2       0.25       0.30       0.27        146
     3       0.20       0.23       0.22        151
     4       0.23       0.26       0.25        137
     5       0.12       0.12       0.12        126
     6       0.36       0.24       0.29        168
     7       0.37       0.30       0.33        149
     8       0.43       0.46       0.44        165
     9       0.39       0.25       0.31        157

   accuracy                    0.30        1500
  macro avg                    0.30        1500
weighted avg                    0.31        1500

time: 132.4669478
```

A

$\gamma=0.001$

```
accuracy: 0.298
precision: [0.27918782 0.38607595 0.24581006 0.20231214 0.23076923 0.12
0.36283186 0.3697479 0.42937853 0.38834951]
recall: [0.36912752 0.40131579 0.30136986 0.23178808 0.26277372 0.11904762
0.24404762 0.29530201 0.46060606 0.25477707]
      precision    recall  f1-score   support

     0       0.28       0.37       0.32        149
     1       0.39       0.40       0.39        152
     2       0.25       0.30       0.27        146
     3       0.20       0.23       0.22        151
     4       0.23       0.26       0.25        137
     5       0.12       0.12       0.12        126
     6       0.36       0.24       0.29        168
     7       0.37       0.30       0.33        149
     8       0.43       0.46       0.44        165
     9       0.39       0.25       0.31        157

   accuracy                    0.30        1500
  macro avg                    0.30        1500
weighted avg                    0.31        1500

time: 120.021547
```

Παρατηρούμε καλύτερη επίδοση για $\gamma=1$, και κρατάμε αυτό ενώ δοκιμάζουμε άλλες παραμέτρους:

C=10

```
accuracy: 0.298
precision: [0.27918782 0.38607595 0.24581006 0.20231214 0.23076923 0.12
0.36283186 0.3697479 0.42937853 0.38834951]
recall: [0.36912752 0.40131579 0.30136986 0.23178808 0.26277372 0.11904762
0.24404762 0.29530201 0.46060606 0.25477707]
      precision  recall  f1-score  support
0      0.28      0.37      0.32      149
1      0.39      0.40      0.39      152
2      0.25      0.30      0.27      146
3      0.20      0.23      0.22      151
4      0.23      0.26      0.25      137
5      0.12      0.12      0.12      126
6      0.36      0.24      0.29      168
7      0.37      0.30      0.33      149
8      0.43      0.46      0.44      165
9      0.39      0.25      0.31      157

      accuracy
macro avg      0.30      0.29      0.29      1500
weighted avg    0.31      0.30      0.30      1500

time: 188.1691008
```

C=1

```
accuracy: 0.30333333333333334
precision: [0.29319372 0.3961039 0.25280899 0.19753086 0.22641509 0.14074074
0.37931034 0.35772358 0.43181818 0.39622642]
recall: [0.37583893 0.40131579 0.30821918 0.21192053 0.26277372 0.15079365
0.26190476 0.29530201 0.46060606 0.26751592]
      precision  recall  f1-score  support
0      0.29      0.38      0.33      149
1      0.40      0.40      0.40      152
2      0.25      0.31      0.28      146
3      0.20      0.21      0.20      151
4      0.23      0.26      0.24      137
5      0.14      0.15      0.15      126
6      0.38      0.26      0.31      168
7      0.36      0.30      0.32      149
8      0.43      0.46      0.45      165
9      0.40      0.27      0.32      157

      accuracy
macro avg      0.31      0.30      0.30      1500
weighted avg    0.31      0.30      0.30      1500

time: 161.094609
```

Καλύτερη επίδοση έχουμε για το συνδυασμό $\gamma=C=1$.

Τώρα ας δοκιμάσουμε να αλλάξουμε τη συνάρτηση kernel σε rbf. Το accuracy που βρέθηκε είναι 0.11 οπότε απορρίπτουμε κατευθείαν αυτή την περίπτωση.

Καταλήγουμε, λοιπόν, ότι το ιδανικό μοντέλο είναι αυτό με παραμέτρους: kernel: linear, $\gamma=1$, $C=1$ και το εκπαιδεύουμε σε μεγαλύτερο αριθμό δεδομένων του σετ. (train: 10000, test: 2000)

```
accuracy: 0.3368
precision: [0.29581994 0.43277311 0.26041667 0.21505376 0.23508772 0.28436019
0.39631336 0.37914692 0.48797251 0.4556213 ]
recall: [0.368 0.42386831 0.29296875 0.24793388 0.28033473 0.24793388
0.32330827 0.33898305 0.52985075 0.29844961]
      precision    recall  f1-score   support

0         0.30        0.37        0.33         250
1         0.43        0.42        0.43         243
2         0.26        0.29        0.28         256
3         0.22        0.25        0.23         242
4         0.24        0.28        0.26         239
5         0.28        0.25        0.26         242
6         0.40        0.32        0.36         266
7         0.38        0.34        0.36         236
8         0.49        0.53        0.51         268
9         0.46        0.30        0.36         258

   accuracy                   0.34        2500
  macro avg                   0.34        0.34        0.34        2500
weighted avg                   0.35        0.34        0.34        2500

time: 284.83180480000004
```

Με άλλα λόγια, ο κώδικας είχε περισσότερη επιτυχία στο να ταξινομεί πλοία και φορτηγά, και χειρότερη επίδοση στο να αναγνωρίζει γάτες.

Ο κώδικας που χρησιμοποιήθηκε, δεδομένου ότι οι παραλλαγές χρειάζονται μόνο αλλαγή στην εντολή

```
cls=svm.SVC(kernel="rbf", gamma=1, C=1)
```

είναι ο εξής:

```

1  from sklearn import svm
2  from keras.datasets import cifar10
3  from keras.utils import np_utils
4  from sklearn import metrics
5  import timeit
6
7  (x_train, y_train) , (x_test, y_test)=cifar10.load_data()
8
9  x_train=x_train.reshape(50000,3072)
10 x_test=x_test.reshape(10000,3072)
11
12 x_train=x_train[:10000, :]
13 x_test=x_test[:2500, :]
14 y_train=y_train[:10000]
15 y_test=y_test[:2500]
16
17 x_train=x_train/255
18 x_test=x_test/255
19
20 cls=svm.SVC(kernel="linear", gamma=1, C=1)
21 t_start=timeit.default_timer()
22 cls.fit(x_train,y_train)
23 pred=cls.predict(x_test)
24 t_stop=timeit.default_timer()
25 print("accuracy: ", metrics.accuracy_score(y_test,pred))
26 print("precision: ", metrics.precision_score(y_test,pred, average= None))
27 print("recall: ", metrics.recall_score(y_test, pred, average=None))
28 print(metrics.classification_report(y_test,pred))
29 print("time: ", t_stop-t_start)
30

```

Τώρα περνάμε στην υλοποίηση του αλγορίθμου K-Nearest-Neighbors, με τον κώδικα που επισυνάπτεται στον φάκελο:

K=1

```

classification report:
              precision    recall  f1-score   support

     0       0.30      0.38      0.34      250
     1       0.50      0.15      0.23      243
     2       0.21      0.35      0.26      256
     3       0.23      0.17      0.19      242
     4       0.20      0.36      0.25      239
     5       0.27      0.20      0.23      242
     6       0.26      0.31      0.28      266
     7       0.44      0.20      0.27      236
     8       0.40      0.59      0.48      268
     9       0.55      0.12      0.19      258

 accuracy          0.29      2500
 macro avg         0.33      0.28      0.27      2500
 weighted avg      0.34      0.29      0.28      2500

 accuracy: {0.2864: 0.4}
 time: 1.5358133

```

K=3

```

classification report:
              precision    recall  f1-score   support

     0       0.26      0.50      0.34      250
     1       0.50      0.19      0.27      243
     2       0.18      0.46      0.26      256
     3       0.22      0.18      0.20      242
     4       0.18      0.32      0.23      239
     5       0.30      0.11      0.16      242
     6       0.34      0.23      0.27      266
     7       0.63      0.13      0.22      236
     8       0.49      0.56      0.52      268
     9       0.75      0.08      0.15      258

 accuracy          0.28      2500
 macro avg         0.39      0.28      0.26      2500
 weighted avg      0.39      0.28      0.27      2500

 accuracy: {0.2796: 0.4}
 time: 2.0811693

```

Είναι εμφανές ότι με τη μέθοδο των πλησιέστερων γειτόνων έχουμε μεγάλα ποσοστά accuracy στα φορτηγά (0.5 και 0.75) αλλά δυσκολεύεται ιδιαίτερα στην αναγνώριση ταράνδων.

Αν και σαφώς γρηγορότερος του SVM, δεν έχει το ίδιο καλές τιμές ακρίβειας.

Τέλος, ο αλγόριθμος Nearest Centroids:

Classification report:				
	precision	recall	f1-score	support
0	0.24	0.49	0.32	250
1	0.31	0.22	0.26	243
2	0.24	0.09	0.14	256
3	0.36	0.06	0.11	242
4	0.19	0.07	0.10	239
5	0.25	0.26	0.25	242
6	0.25	0.60	0.35	266
7	0.27	0.19	0.22	236
8	0.47	0.40	0.43	268
9	0.36	0.44	0.40	258
accuracy			0.29	2500
macro avg	0.29	0.28	0.26	2500
weighted avg	0.30	0.29	0.26	2500
accuracy: 0.2876				
time: 0.23920969999999997				

Επίσης σαφώς γρηγορότερος του SVM, αλλά επίσης όχι με εξίσου καλές τιμές ακρίβειας. Καλύτερη επίδοση έχει στην αναγνώριση πλοίων.