Αναφορά στην 3η εργασία
Όραση Υπολογιστών

Χανιώτης Παναγιώτης
57636

05 – 12 - 2021

Χανιώτης Παναγιώτης
57636

Περιεχόμενα

Θεωρητική Προσέγγιση Σελίδα 3

Ανάλυση του προγράμματος Σελίδα 8

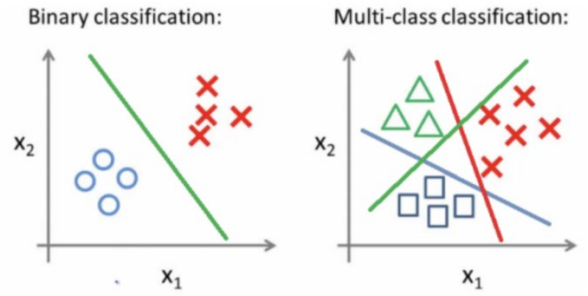
Βιβλιογραφία Σελίδα 15

Θεωρητική Προσέγγιση

Χανιώτης Παναγιώτης
57636

Ζητούμενο της παρούσας εργασίας είναι η ταξινόμηση πολλαπλών κλάσεων. Αυτό θα γίνει κάνοντας χρήση δύο αλγορίθμων. Τον αλγόριθμο k-NN και τον αλγόριθμο one-versus-all χρήσει ενός SVM ταξινόμητη. Τέλος καλούμαστε να αξιολογήσουμε τα αποτελέσματα εκφράζοντας ένα ποσοστό επιτυχίας σε κάθε περίπτωση.

Στη μηχανική μάθηση, η ταξινόμηση πολλαπλών κλάσεων είναι το πρόβλημα της ταξινόμησης παρουσιών σε μία από τρεις ή περισσότερες κλάσεις. Η ταξινόμηση των παρουσιών σε μία από δύο κλάσεις ονομάζεται δυαδική ταξινόμηση.



Οι τεχνικές που αναπτύχθηκαν με βάση τη μείωση του προβλήματος πολλαπλών κλάσεων σε πολλαπλά δυαδικά προβλήματα μπορούν επίσης να ονομαστούν τεχνικές μετασχηματισμού προβλήματος. Παραδείγματα τέτοιων τεχνικών είναι οι:

One-vs.-all (ένα εναντίων όλων)

Αυτή περιλαμβάνει την εκπαίδευση ενός μόνο ταξινομητή ανά τάξη, με τα δείγματα αυτής της κατηγορίας ως θετικά δείγματα και όλα τα άλλα δείγματα ως αρνητικά. Αυτή η στρατηγική απαιτεί από τους βασικούς ταξινομητές να παράγουν μια βαθμολογία εμπιστοσύνης πραγματικής αξίας για την απόφασή τους και όχι απλώς μια ετικέτα κλάσης. Οι διακριτές ετικέτες κλάσεων από μόνες τους μπορούν να οδηγήσουν σε ασάφειες, όπου προβλέπονται πολλαπλές κλάσεις για ένα μόνο δείγμα.

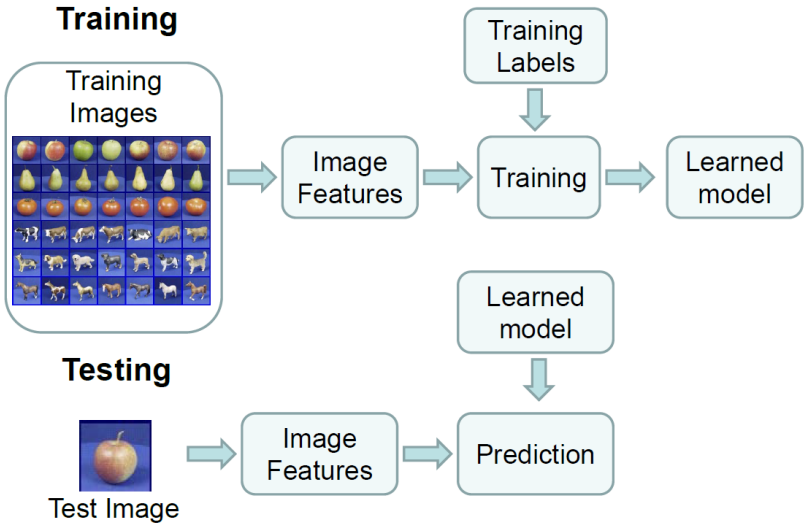
One-vs.-one (ένα εναντίων ενός)

Στην αναγωγή ενός έναντι ενός, κάποιος εκπαιδεύει K (K − 1) / 2 δυαδικούς ταξινομητές για ένα πρόβλημα πολλαπλών κλάσεων K-way. καθένας λαμβάνει τα δείγματα ενός ζεύγους τάξεων από το αρχικό σετ εκπαίδευσης και πρέπει να μάθει να διακρίνει αυτές τις δύο τάξεις.

Από το σημείο που θα μετατρέψουμε το πρόβλημα μας σε διάδικό, έχουμε πλέον την δυνατότητα να εφαρμόσουμε αρκετούς αλγορίθμους. Αυτοί μπορούν να βασίζονται σε νευρωνικά δίκτυα, δέντρα αποφάσεων**, k-πλησιέστερους γείτονες** (k-means), αφελείς Bayes, **μηχανές υποστήριξης διανυσμάτων** (Support Vector Machines) και μηχανές ακραίας μάθησης για την αντιμετώπιση προβλημάτων ταξινόμησης πολλαπλών κλάσεων.

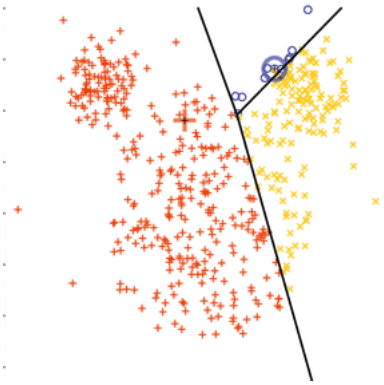
Χανιώτης Παναγιώτης
57636

Για να εφαρμόσουμε κάποιον από τους δύο αλγορίθμους που μας ενδιαφέρουν θα πρέπει αρχικά να λάβουμε χαρακτηριστικά από την κάθε εικόνα, να δημιουργήσουμε ένα λεξικό από αυτά και να δημιουργήσουμε τις κλάσεις που δηλώνουν την εκάστοτε ομάδα στην οποία ανήκει η κάθε μια από αυτές. Το διάνυσμα με τα χαρακτηριστικά τους θα υπολογιστεί με τον αλγόριθμο **SIFT.**



Η παραγωγή αυτού του λεξικού γίνεται με τον αλγόριθμο k-means (ή αλγόριθμο Lloyd). Για να καταλάβουμε πως λειτουργεί, αρκεί να φανταστούμε ότι κάθε πιθανός περιγραφέας έχει 2 τιμές που τον περιγράφουν, άρα βρίσκεται στο επίπεδο (ο αλγόριθμος SIFT επιστρέφει διανύσματα μεγέθους 128, επομένως στην πραγματικότητα όλα αυτά συμβαίνουν σε έναν υπερχώρο πολλαπλών διαστάσεων – δηλαδή πρακτικά μη παραστάσιμο γραφικά).

Ο αλγόριθμος αρχικα ορίζει αυθέρετα σημεία ο αριθμός των οποίον εξαρτάται από το πόσες κλάσεις αναζητούμε.

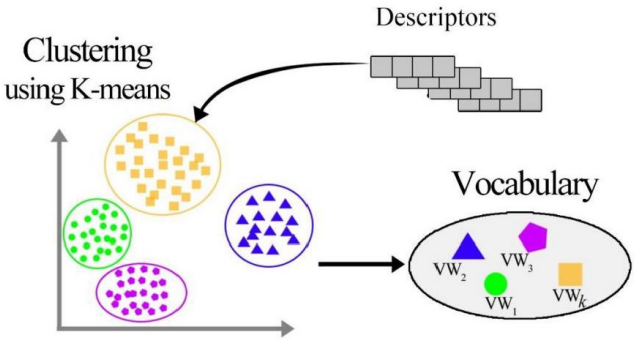


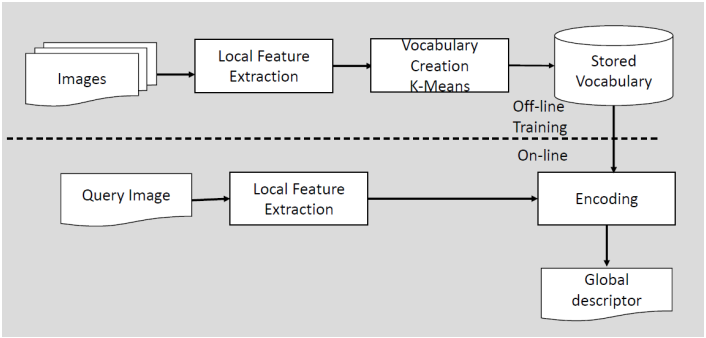
Στη συνέχεια, προβαίνει σε μετρήσεις όλων των σημείων από τα κέντρα τους. Μετά ομαδοποιείται το κάθε σημείο ανάλογα την απόσταση που έχει από το κάθε κέντρο. Τέλος, κάθε υπολογίζεται ο μέσος όρος της κάθε ομάδας και τοποθετείται στον χώρο ως

Χανιώτης Παναγιώτης
57636

το νέο κέντρο. Με τα νέα κέντρα για κάθε μετρούμενη κλάση επαναλαμβάνουμε τον αλγόριθμο μέχρι η σχέση των παλιών με νέα κέντρα, δηλαδή η μετακίνηση τους να είναι πολύ μικρή.

Κάθε κλάση μπορούμε να την χαρακτηρίσουμε από το κέντρο της. Όλα τα κέντρα τοποθετούνται σε ένα λεξικό.



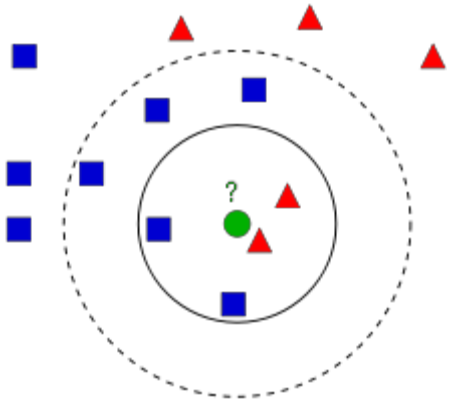


Χανιώτης Παναγιώτης
57636

Ο αλγόριθμος k-NN

Είναι ένας από τους απλούστερους αλγόριθμους ταξινόμησης που είναι διαθέσιμοι για εποπτευόμενη μάθηση. Η ιδέα είναι η αναζήτηση της πλησιέστερης αντιστοίχιση των δεδομένων δοκιμής στον χώρο των χαρακτηριστικών. Κάθε νέος descriptor εικόνας την οποία εξετάζουμε θα συγκριθεί με ομαδοποιημένα descriptors από το λεξικό και θα αποφανθεί σε ποια ομάδα να το τοποθετήσει βάσει της πλησιέστερης σε αυτό ομάδας. Ο συντελεστής k είναι μια παράμετρος του αλγορίθμου η οποία δηλώνει πόσοι είναι οι πλησιέστεροι γείτονες.

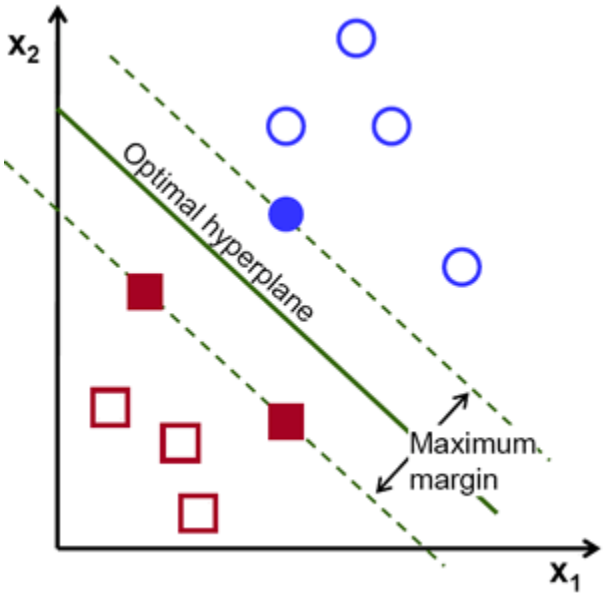
Το θετικό του αλγορίθμου είναι ότι απαιτεί λίγη εκπαίδευση. Το αρνητικό του είναι ο μεγαλός αριθμός πράξεων που απαιτεί, επειδή θα πρέπει να αφαιρέσουμε το σημείο που εξετάζουμε με κάθε άλλο σημείο στο χώρο για να καταλήξουμε στην ελάχιστη απόσταση.



Χανιώτης Παναγιώτης
57636

Ο αλγόριθμος χρήσει SVM

Ο αλγόριθμος SVM απαιτεί την εκπαίδευση του συστήματος χωρίζοντας τις κλάσεις μεταξύ τους με υπερεπίπεδα. Η αναπαράσταση τους γίνεται με προβολή των πολυδιάστατων descriptors στις δύο διαστάσεις.



Χανιώτης Παναγιώτης
57636

Ανάλυση του προγράμματος

Αρχικά, καλούμαστε να εκπαιδεύσουμε το σύστημα μας χρησιμοποιώντας τις εικόνες εκπαίδευσης (training) ώστε να παράξουμε ένα λεξικό. Σύμφωνα με τις απαιτήσεις τις εργασίας, θα υπάρχει ένας κώδικας που θα χρησιμοποιεί τον αλγόριθμο KNN και ένας για τον αλγόριθμο SVM. Για να το κάνουμε αυτό θα πρέπει να ακολουθήσουμε τα παρακάτω βήματα.

Η συνάρτηση *create_volcabulary* αντλεί τα τοπικά χαρακτηριστικά τις κάθε εικόνας χρησιμοποιώντας τον αλγόριθμο SIFT. Όλα τα χαρακτηριστικά από την κάθε εικόνα του κάθε αποθηκεύονται στον πίνακα *train_descs*. Κάθε φορά που προσθέτοντας καινούργια στοιχεία σε αυτόν, ο πίνακας μεγαλώνει χρησιμοποιώντας την μέθοδο concatenate. Στη συνέχεια, δημιουργούμε το λεξικό με τον αλγόριθμο k-means. Κριτήριο τερματισμού του αλγορίθμου είναι είτε οι 30 επαναλήψεις της διαδικασίας είτε η πολύ μικρή μετακίνηση του κέντρου των κλάσεων. Τέλος αποθηκεύουμε το αποτέλεσμα σε αρχείο numpy. (Σημείωση: μας ενδιαφέρει μόνο η τρίτη έξοδος της μεθόδου *cv.kmeans)*

```python
def create_vocabulary(train_folders):
    print('Extracting features...')
    train_descs = np.zeros((0, 128))
    for folder in train_folders:
        files = os.listdir(folder)
        for file in files:
            path = os.path.join(folder, file)
            desc = extract_local_features(path)
            if desc is None: # For Thumbs.db
                continue
            train_descs = np.concatenate((train_descs, desc), axis=0)

    # Create vocabulary
    print('Creating vocabulary...')
    term_crit = (cv.TERM_CRITERIA_EPS, 30, 0.1)
    vocabulary = cv.kmeans(train_descs.astype(np.float32), 50, None, term_crit, 1, 0)[2]
    np.save('vocabulary.npy', vocabulary)
    return vocabulary
```

Η συνάρτηση *create_index* δημιουργεί ένα αρχείο index που περιέχει το ιστόγραμμα *bovw*. Η διαδικασία που ακολουθείται εδώ ακολουθεί πάλι την άντληση των τοπικών χαρακτηριστικών. Η κωδικοποίηση πραγματοποιείται στην συνάρτηση *encode_bovw_descriptor*. Η αποθήκευση του αρχείου γίνεται με μεθόδους της βιβλιοθήκης json.

```python
def create_index(train_folders, vocabulary):
    # Create Index
    print('Creating index...')
    img_paths = []
    bow_descs = np.zeros((0, vocabulary.shape[0]))
    for folder in train_folders:
        files = os.listdir(folder)
        for file in files:
            path = os.path.join(folder, file)
            desc = extract_local_features(path)
            if desc is None:
                continue
            bow_desc = encode_bovw_descriptor(desc, vocabulary)

            img_paths.append(path)
            bow_descs = np.concatenate((bow_descs, bow_desc), axis=0)

    np.save('index.npy', bow_descs)
    with open('index_paths.txt', mode='w+') as file:
        json.dump(img_paths, file)
    return img_paths, bow_descs
```

Η υλοποίηση συνάρτηση *encode_bovw_desciptor* είναι ως εξής · σε

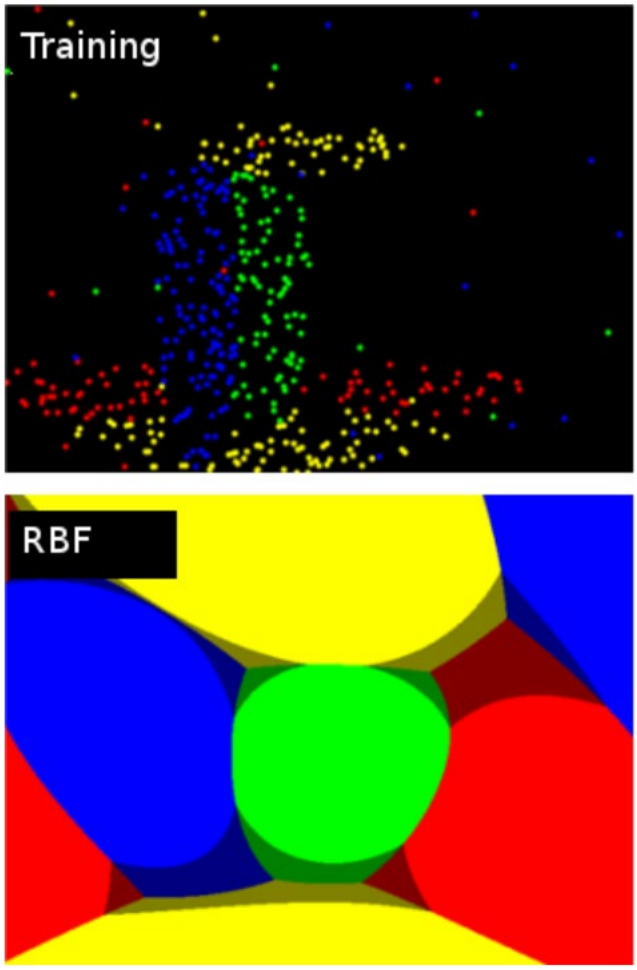
```python
def encode_bovw_descriptor(desc, vocabulary):
    bow_desc = np.zeros((1, vocabulary.shape[0]))
    for d in range(desc.shape[0]):
        distances = np.sum((desc[d, :] - vocabulary) ** 2, axis=1)
        mini = np.argmin(distances)
        bow_desc[0, mini] += 1
    if np.sum(bow_desc) > 0:
        bow_desc = bow_desc / np.sum(bow_desc)
    return bow_desc
```

Η εκπαίδευση του SVM γίνεται χρήσει ενός αντικειμένου svm το οποίο δημιουργούμε με διάφορα χαρακτηριστικά βάσει των οποίων θα γίνει η εκτέλεση του αλγορίθμου svm. Οι παράμετροι επιλέχτηκαν προκειμένου να εκτελεστεί βέλτιστα για την περίπτωση μας.

Αρχικά ορίζουμε τον τύπου SVM ως C_SVC. Αυτό είναι για την κατηγοριοποίηση (classification) ομάδων μεγαλύτερων ίσων του 2. Αυτός ο

τύπος είναι ιδανικός για διαχωρισμό μη ιδανικών κλάσεων, δηλαδή με μη γραμμικά διαχωρίσιμα δεδομένα εκπαίδευσης.

Ο τύπος του πύρινα (kernel) επηρεάζει τον τρόπο που θα διαμορφώσουμε τα δεδομένα εισόδου για να τα φέρουμε σε τέτοια μορφή ώστε να είναι πιο εύκολος ο γραμμικός διαχωρισμός τους. Η παρακάτω εικόνα μας δίνει μια εικόνα του πως φαίνεται αυτή η μετατροπή των δεδομένων εκπαίδευσης αν θεωρήσουμε ότι περιγράφονται με μόνο δυο χαρακτηριστικά, άρα και αναπαρίστανται στις δύο διαστάσεις. Ο πυρήνας που επιλέγουμε είναι ο RBF (Radial basis function). Η συνάρτηση RBF είναι μια συνάρτηση με πραγματική τιμή φ της οποίας η τιμή εξαρτάται μόνο από την απόσταση μεταξύ της εισόδου και κάποιου σταθερού σημείου, είτε από την αρχή είτε από ένα σημείο c.



Το κριτήριο τερματισμού της επαναληπτικής διαδικασίας εκπαίδευσης SVM που επιλύει μια μερική περίπτωση περιορισμένου τετραγωνικού προβλήματος βελτιστοποίησης.

Χανιώτης Παναγιώτης
57636

Η μέθοδος *trainAuto* ξεκινάει την διαδικασία της εκπαίδευσης. Η μέθοδος save αποθηκεύει το πλέον εκπαιδευμένο σύστημα.

```
labels = []
for p in img_paths:
    labels.append('145.motorbikes-101' in p)
labels = np.array(labels, np.int32)

print('Training Motorbike SVM...')
svm = cv.ml.SVM_create()
svm.setType(cv.ml.SVM_C_SVC)
svm.setKernel(cv.ml.SVM_RBF)
svm.setTermCriteria((cv.TERM_CRITERIA_COUNT, 100, 1.e-06))
svm.trainAuto(bow_descs.astype(np.float32), cv.ml.ROW_SAMPLE, labels)
svm.save('svm_motorbike')
```

Επαναλαμβάνουμε την διαδικασία και για τις υπόλοιπες εικόνες.

Για τον έλεγχο θα πρέπει να γράψουμε κώδικες που θα κάνουν test. Αρχικά θα αναλύσουμε την υλοποίηση του αλγορίθμου KNN. Δύο επιπλέον συναρτήσεις που θα χρειαστούμε είναι οι παρακάτω. Αυτές φορτώνουν στο πρόγραμμα τις αποθηκευμένες τιμές του λεξικού και του index αντίστοιχα. Τέλος αναπτύχθηκε μια επιπλέον με σκοπό να εξαγάγει το ποσοστό επιτυχίας

```
def load_vocabulary():
    vocabulary = np.load('vocabulary.npy')
    return vocabulary

def load_index():
    # Load Index
    bow_descs = np.load('index.npy')
    with open('index_paths.txt', mode='r') as file:
        img_paths = json.load(file)
    return img_paths, bow_descs
```

```
def print_accuracy(correct, num):
    if num == 0:
        print("run test first!")
    else:
        h = (correct / num) * 100
        print("Correct percentage is " + str(h) + "% \n")
```

Για την αναγνώριση με KNN θα ορίσουμε αρχικά την αντίστοιχη συνάρτηση. Θα χρησιμοποιηθούν οι μεταβλητές *correct* και *frequency* οι οποίες μας δίνουν πληροφορίες για το πλήθος των σωστών ευρέσεων καθώς και την κατανομή ανά είδος εικόνας. Οι μέθοδοι και οι συναρτήσεις που χρησιμοποιούνται είναι ίδιες με αυτές της εκπαίδευσης. Γνωρίζοντας το όνομα του

αρχείου, το οποίο περιλαμβάνει το είδος του αντικειμένου, μας επιτρέπει, χρήσει των regular expressions να ελέγξουμε αν το αποτέλεσμα που παίρνουμε είναι σωστό ώστε να μπορέσουμε να εξάγουμε στατιστικά στοιχεία επιτυχίας του προγράμματος.

```
def kNN(query_img_path, N):
    global correct
    frequency = np.zeros(5)
    desc = extract_local_features(query_img_path)
    bow_desc = encode_bovw_descriptor(desc, vocabulary)
    distances = np.sum((bow_desc - bow_descs) ** 2, axis=1)
    ids = np.argsort(distances)
    ids = retrieved_ids[0:N]
    for id in ids.tolist():
        if re.search('.*motorbike.*', img_paths[id]):
            frequency[0] += 1
        elif re.search('.*bus.*', img_paths[id]):
            frequency[1] += 1
```

```
    index_max = np.argmax(frequency)
    if index_max == 0:
        if re.search('.*motorbike.*', folder):
            correct += 1
    if index_max == 1:
        if re.search('.*bus.*', folder):
            correct += 1
```

Ο τελευταίος κώδικας αναλύει την δοκιμή (test) με την μέθοδο svm.

Πρώτα χρειαζόμαστε ένα αντικείμενο descriptor_extractor το οποίο θα περιλαμβάνει το ιστόγραμμα Bag of Words (κεντρικό descriptor). Το δημιουργούμε με την

συνάρτηση constructor cv.BOWImgDescriptroExtactor. Σε αυτή θα δώσουμε ως όρισμα το αντικείμενο sift και την παράμετρο τον BFMatcher, δηλαδή για κάθε περιγραφέα στο πρώτο σετ, αυτό το ταίριασμα βρίσκει τον πλησιέστερο περιγραφέα στο δεύτερο σετ δοκιμάζοντας το καθένα. Αυτή η αντιστοίχιση περιγραφέων υποστηρίζει απόκρυψη επιτρεπόμενων αντιστοιχιών συνόλων περιγραφών.

Το μέγεθος ή νόρμα (norm) που χρησιμοποιούμε έχει την παρακάτω μαθηματική περιγραφή

$$norm = \begin{cases} \|\mathtt{src1}\|_{L_2}^2 = \sum_I \mathtt{src1}(I)^2 & \text{if } \mathtt{normType} = \mathtt{NORM\_L2SQR} \\ \|\mathtt{src1} - \mathtt{src2}\|_{L_2}^2 = \sum_I (\mathtt{src1}(I) - \mathtt{src2}(I))^2 & \text{if } \mathtt{normType} = \mathtt{NORM\_L2SQR} \\ \left(\frac{\|\mathtt{src1} - \mathtt{src2}\|_{L_2}}{\|\mathtt{src2}\|_{L_2}} \right)^2 & \text{if } \mathtt{normType} = \mathtt{NORM\_RELATIVE} \mid \mathtt{NORM\_L2SQR} \end{cases}$$

```
# Classification
descriptor_extractor = cv.BOWImgDescriptorExtractor(sift, cv.BFMatcher(cv.NORM_L2SQR))
descriptor_extractor.setVocabulary(vocabulary)



# Load SVM
svm_motorbike = cv.ml.SVM_create()
svm_motorbike = svm_motorbike.load('svm_motorbike')

svm_bus = cv.ml.SVM_create()
svm_bus = svm_bus.load('svm_bus')
```

Τα αποτελέσματα για το σε ποια κλάση είναι τοποθετημένη η ερωτηθείσα εικόνα τα παίρνουμε από την μέθοδο predict του αντικειμένου svm. Εδώ ξεκαθαρίζουμε τον τύπο

```
response_motorbike = svm_motorbike.predict(bow_desc.astype(np.float32), flags=cv.ml.STAT_MODEL_RAW_OUTPUT)
```

Το επαναλαμβάνουμε για όλες τις κλάσεις και αφού τοποθετίσουμε όλα τα αποτελέσματα (τη δευτερη εξοδο της μεθοδου predict) σε ένα πίνακα, βρίσκουμε την ελάχιστη τιμή που δηλώνει την ελάχιστη απόσταση από το υπερεπίπεδο SVM

```
#Smallest Distance
resp_array = np.array([response_motorbike[1], response_bus[1], response_bike[1], response_airplane[1]
minIndex = np.argmin(resp_array)
```

Έξοδος για KNN:

```
For i = 6 nearest neighboors
Correct percentage is 69.23076923076923%

For i = 11 nearest neighboors
Correct percentage is 63.46153846153846%
```

Έξοδος για SVM:

```
For i = 6 nearest neighboors
Correct percentage is 69.23076923076923%

For i = 11 nearest neighboors
Correct percentage is 63.46153846153846%
```


Χανιώτης Παναγιώτης
57636

Βιβλιογραφία

Computer Vision Algorithms and Applications, Richard Szeliski

Computer Vision Lectures, Ioannis Pratikakis

Computer Vision Lab Course, Lazaros Tsochatzidis

https://docs.opencv.org/3.4/d1/d73/tutorial_introduction_to_svm.html

https://docs.opencv.org/3.4/d1/d2d/classcv_1_1ml_1_1SVM.html#aad7f1aaccced3c33bb256640910a0e56

https://docs.opencv.org/3.4/d3/da1/classcv_1_1BFMatcher.html#details

https://docs.opencv.org/4.x/d5/d26/tutorial_py_knn_understanding.html

https://docs.opencv.org/4.x/de/d4d/tutorial_py_kmeans_understanding.html