Boğaziçi University

CMPE537 Assignment 3
Image Classification with Bag-of-Features

Elif Çalışkan

2016400183

December 8, 2019

0.1 Methodology:

0.1.1 Compute SIFT descriptors:

First, I iterate through every file in train directory, read and resize each image in directories: city, face, green, house_building, house_indoor, office, sea. After reading those files, using sift.detectAndCompute(img, None) function, I find sift descriptors. These descriptors have size ax128(a varies in different images). These descriptors are appended to descriptions array for kmeans. Directory name is appended to labels array.

0.1.2 Find the dictionary:

After finding the descriptions 2d array, kmeans clustering is performed with 50 clusters. With the help of kmeans.predict() function, every description array's cluster value is predicted. The created array have values between 0 and clusterCount.

0.1.3 Feature quantization:

image_descriptions array is created with dimensions filePaths.size x clusterCount. Histogram of each predict array is added to image_descriptions array.

0.1.4 Classification:

With a chi2_kernel, a svc classifier is created and image_descriptions array and labels are fitted. Then for the testing part, each file in test directory is resized and read. Their descriptions are added to test_descriptions 2d array and their labels are stored in test_labels array. Using svc_classifier.predict() function, test prediction array is created. Confusion matrix, classification report and score is found for chi2_kernel classifier. Labels array has values: ['city' 'face' 'green' 'house_building' 'house_indoor' 'office' 'sea' 'sea']

0.2 Comments:

Confusion matrix:

9	0	21	0	0	0	0
0	2	28	0	0	0	0
0	0	30	0	0	0	0
0	0	28	2	0	0	0
0	0	30	0	0	0	0
0	0	20	0	0	10	0
0	0	23	0	0	0	7

Classification report

	precision	recall	f1-score	support
city	1.00	0.30	0.46	30
face	1.00	0.07	0.12	30
green	0.17	1.00	0.29	30
house_building	1.00	0.07	0.12	30
house_indoor	0.00	0.00	0.00	30
office	1.00	0.33	0.50	30
sea	1.00	0.23	0.38	30
micro avg	0.29	0.29	0.29	210
macro avg	0.74	0.29	0.27	210
weighted avg	0.74	0.29	0.27	210

A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known. It allows the visualization of the performance of an algorithm. We can see that the values except the diagonal, has high values so we can say the the performance of this classification is not good. Score is found with the function svc_classifier.score(test_descriptions, test_labels) and the value is 0.2857142857142857.

test_labels has values: ['green' 'city' 'city' 'green' 'green' 'green' 'green' 'city' 'city' 'city' 'city' 'green' 'green' 'green' 'green' 'green' 'green' 'green' 'green' 'green' 'green' 'green' 'green' 'city' 'city' 'green' 'city' 'green'

'green' 'green' 'green' 'green' 'green' 'green' 'green' 'green' 'green' 'green' 'face' 'face' 'green' 'house_building' 'house_building' 'green' 'office' 'office' 'office' 'office' 'office' 'office' 'office' 'office' 'office' 'office' 'green' 'green' 'green' 'green' 'green' 'green' 'green' 'green' 'green' 'green' 'green' 'green' 'green' 'green' 'green' 'green' 'green' 'green' 'sea' 'sea' 'sea' 'green' 'green' 'green' 'green' 'sea' 'green' 'green' 'green' 'green' 'sea' 'sea' 'green' 'green' 'green' 'green' 'green' 'green' 'green' 'green' 'green' 'green' 'green' 'green' 'green' 'sea']

Then, with gridSearch, best estimate parameters are found: C = 10, kernel = linear. The best score with these parameters is 0.573729863692689. A new classifier is created with chosen parameters. With that new classifier, test_descriptions are predicted and their labels are compared to the target. The score of this classification is 0.5714285714285714. We can clearly see that the score is higher but still not equal to the best score. Since gridSearch tries every configuration and finds the best estimate, the performance is improved with given parameters: C = 10 and kernel = linear.

This predictions array has values: ['green' 'city' 'city' 'sea' 'city' 'office' 'city' 'city' 'city' 'city' 'city' 'green' 'city' 'green' 'city' 'green' 'green' 'city' 'green' 'city' 'city' 'city' 'city' 'city' 'city' 'city' 'house_building' 'city' 'green' 'city' 'face' 'face' 'face' 'face' 'face' 'house_indoor' 'face' 'face' 'face' 'face' 'house_indoor' 'face' 'face' 'green' 'face' 'face' 'face' 'face' 'face' 'face' 'house_indoor' 'face' 'face' 'green' 'face' 'face' 'city' 'green' 'face' 'face' 'green' 'sea' 'face' 'green' 'green' 'city' 'green' 'green' 'city' 'green' 'green' 'green' 'green' 'house_indoor' 'green' 'green' 'house_indoor' 'green' 'city' 'green' 'green' 'face' 'house_indoor' 'green' 'green' 'green' 'green' 'city' 'green' 'city' 'green' 'house_building' 'house_building' 'house_building' 'office' 'house_building' 'house_indoor' 'city' 'office' 'house_building' 'city' 'house_building' 'house_building' 'house_building' 'house_building' 'office' 'house_building' 'office' 'city' 'office' 'city' 'office' 'house_building' 'house_building' 'house_indoor' 'city' 'house_building' 'house_building' 'house_building' 'face' 'office' 'office' 'house_indoor' 'office' 'house_building' 'office' 'city' 'house_indoor' 'office' 'office' 'green' 'office' 'office' 'city' 'office' 'office' 'house_indoor' 'house_indoor' 'green' 'house_indoor' 'office' 'house_indoor' 'city' 'face' 'office' 'office' 'house_indoor' 'office' 'office' 'house_indoor' 'house_building' 'house_building' 'office' 'office' 'office' 'office' 'office' 'office' 'office' 'office' 'office' 'office' 'office' 'house_building' 'house_building' 'office' 'green' 'city' 'office' 'city' 'green' 'office' 'office' 'house_indoor' 'house_indoor' 'house_indoor' 'green' 'office' 'office' 'office' 'sea' 'sea' 'sea' 'sea' 'sea' 'sea' 'sea' 'green' 'green' 'sea' 'sea' 'sea' 'face' 'city' 'sea' 'sea' 'sea' 'green' 'sea' 'office' 'green' 'green' 'green' 'green' 'sea' 'sea' 'city' 'face' 'office' 'sea']

For testing the performance of classifier, I used the training dataset. After predicting, the score is found to be 0.8760842627013631. This result is very close to 1 and we can say that the classifier works well. I had a problem finding the confusion matrix of these values even though I followed the same procedure. So, I am not able to add these matrices.

0.3 How to run:

This project doesn't get any arguments, it iterates train and test directories that are in the same path with this project.