Report of the 6th exercise sheet

Marco Adamczyk (MA), Till Brinkmann (TB)

Submission by 17th January 2020

Tutorial: Tuesday 10-12, Tutor: Riza Velioglu

1 Introduction

1.1 Dataset

The dataset consists of 896 pictures in 4 different classes. The pictures are 299 by 299 RGB-pixels in size.
All pictures are nature photographs and seem to be classified as "forest" (class 0, 230 pictures), "stony mountains" (class 1, 412 pictures), "snowy mountains" (class 2, 126 pictures), "lake" (class 3, 128 pictures). Because class 1 has a lot more elements than the other, it could be necessary to balance the dataset. TB

1.2 Initial Features

To get the most important information from the data, a good working feature extraction is needed. It has to abstract the image to a vector, so it can be computed by our algorithm. In this use case it is crucial that the extraction provides very different vectors for each class to ensure a good classification. Our apporach was to split the extraction in two parts. First, the extraction created a vector that contained three elements for each pixel representing the rgb. The highest value of each pixel was denoted with a 1 and the other two with a 0. This approach seemed appropriate because the color is one of the major differences between the classes. On the other hand we took a closer look on the brightness by introducing a hyperparameter as a threshold brightness for each pixel. If the pixel is above the treshold it is denoted as a 1 otherwise as a 0. MA

1.3 Methods/Models

We tried two different versions of the naive bayesian classifier. The first one was a naive bayes we implemented by ourselves. It had a vector with only ones and zeros as input. It computed the output by taking the average of all inputs in each class and chose the class associated with the averaged feature vector having the smallest difference from the input vector. The second classifier is the gaussian naive bayes classifier from the sklearn library. Although it can also handle values other than one and zero, we used the same input for this algorithm. MA

2 Experiments

The given dataset was split into a random trainingsset and a random testset. The testset contained 50 pictures and the trainingsset contained 846 pictures. We first trained the classifier and chose the hyperparameter for the brightness to $(256 * 3.0)/2.0 = 384$.(It is the half of the maximum brightness) After the training, we ran the testset on the classifier and used the classification report method from the sklearn library. It returned the precision, recall and the combination of the two called f1-score. MA

2.1 Results

The following are results of a test with 50 randomly selected datapoints.

	precision	recall	f1-score	support
0 / forest	1.00	0.58	0.74	12
1 / stony mountains	0.70	0.91	0.79	23
2 / snowy mountains	0.57	0.44	0.50	9
3 / lake	0.17	0.17	0.17	6

Results of our classifier.

	precision	recall	f1-score	support
0 / forest	0.91	0.67	0.77	15
1 / stony mountains	0.79	0.96	0.87	24
2 / snowy mountains	0.80	0.80	0.80	5
3 / lake	0.20	0.17	0.18	6

Results of the sklearn classifier.

The tested macro F1-score averaged to 0,65 for both classifiers. Scores on the training set were very similar. TB

3 Additional Features

In the leaves of the forests on the pictures, pixel values change quickly in a small area while for example the snow is very evenly colored. So we compute the difference of every pixel to its neighbours and check if it is above a threshold provided as a hyperparameter. When using a difference threshold of 500 the classifier had a strong bias towards class 1 and did not output other class labels.
Secondly we reordered the data in the "highest rgb" feature vector by colorchannel. This creates a higher similarity between pictures that contain the same colors in different locations. Combined with the old features it gave a higher accuracy overall (0,78 vs 0,65 to 0,7 before) and a little higher f1-score (by 0,05). But since it doubles the size of the feature vector this improvement is not worth the additional compute time. TB

4 Discussion

In general the sklearn GaussianNB had more balanced scores over the classes while our own classifier did better on class 0 and 1 while doing worse on the others. To continue we should experiment more with taking only every n-th pixel or averaging an area of pixels to make feature extraction faster. Especially, the rgb difference computation (described in additional features) is very slow and additionally had to be combined with other features to give acceptable results. But trying the highest-rgb with brightness threshold method on every second pixel almost halfed the macro F1-score. So it is important to find the right method for abbreviating pixels and finding other features to extract information from. TB