Image Categorization

Chao Ni

December 19, 2020

Object recognition is a fundamental problem in computer vision. In this exercise, we will implement two approaches to achieve a simple object recognition task. They are nearest neighbor classification and naive Bayesian classification.

1 Local Feature Extraction

In order to extract feature for each image, we will use histogram counts as an indicator. We will first build a codebook containing all possible patches that an image of car might have. Then for each image, we will find its reference in this codebook and proceed with different classification approaches.

1.1 Feature detection

The first step is to extract feature points that we are interested in. Given the image and the requirement for the grid, we will extract grid points on this image and proceed.

1.2 Feature description

For each grid point, we will use histogram of oriented gradients descriptor. Around the grid point, we will take a small patch. In this exercise, we hard-code the size of the patch, it has 16 cells and each cell has a size of 4 by 4. In each cell we create histograms based on the gradient direction with 8 bins. After that, we concatenate the 16 resulting histograms to produce a 128 dimensional vector for every grid point. Finally, for each image, we will have a couple of grid points (local feature), and each feature has size 128.

2 Codebook Construction

To create the codebook, we will be using training data to build both codebook for cars and for non-cars. Intuitively the codebook can be understood as a technician manual for the car, we will first include all features of all car images altogether, then use kmeans to find clusters among those features. Those clusters are the most representative feature of a car and can be used as reference. Likewise, we do the same thing to the non-car images. A visual result of the codebook can be shown in Fig. 1

3 Bag-of-words image representation

As we already have the dictionary (Codebook), for each image, we now need to find out its position in the dictionary, i.e. to encode this image into a space and proceed on that space. To do so, we first extract the features for this image, and assign each feature (visual word) to a word in the Codebook. We will count the occurrence of all visual words in this image and create a histogram (reference in the dictionary) for it.

4 Nearest Neighbor Classification

For each image, after having its reference in the codebook, we will find the most similar image to it in the training set, if it is more close to a car image, then it will be classified as car, otherwise it would be classified as non-car. This is the basic idea of nearest neighbor classification. In MATLAB, we can simply call **knnsearch** to realize it.

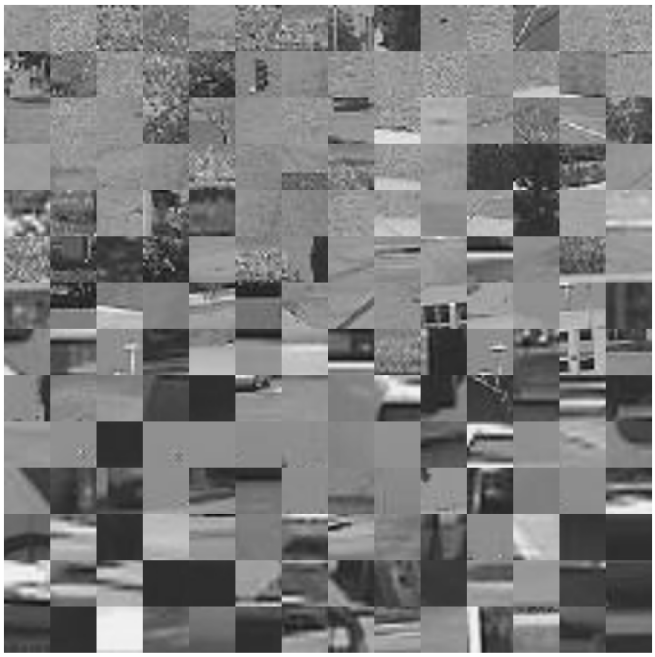

Figure 1: Codebook size 196, as we can see from those patches, some are corresponding to roads and some are corresponding to parts of a car.

5 Bayesian Classification

The difficult part is how to construct a probability scene in this exercise. We are interested in computing the conditional probability given the image $P(Car|hist)$. In the Bayesian scenario, the likelihood is $P(hist|Car)$ and the prior is $P(Car)$. Therefore, we will have $P(Car|hist) \propto P(hist|Car)P(Car)$. The likelihood can be further simplified if we add i.i.d. assumptions to all visual words, i.e. $P(hist|Car) = \prod_{i=1}^{K} P(U(i)|Car)$. For each visual word, we assume a normal distribution, with mean and standard deviation computed from the Bags of words (both for car and non-car images). We assign each image based on its tendency to be a car or not, so if $P(Car|hist) > P(!Car|hist)$, then it would be classified as a car.

6 Discussion

In this section, we will answer the questions in the exercise as well as show some visualization results.

6.1 Classification results

We vary the codebook size from 10 to 300, for each case we run the whole pipeline ten times to eliminate the effect of randomness. We report the highest accuracy here, for NN, we achieve an accuracy of **0.9444** with standard deviation **0.0258**. For Bayesian, we achieve an accuracy of **0.9697** with standard deviation **0.0172**. We can see from Fig. 2 which codebook size are chosen to achieve the best performance respectively. In our case, both performance are achieved when codebook size is 25. We can see Bayes classification is slightly better than NN approach. When the codebook size is rather small, the Bayesian and i.i.d. assumption are still kept, therefore, Bayesian classification can explore more information out of the dataset to achieve slightly better performance.

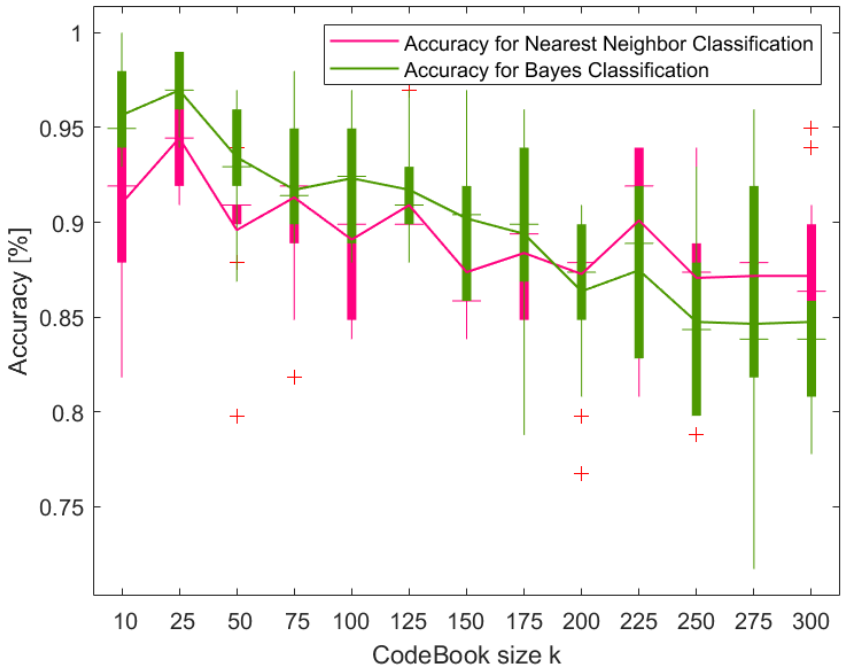

Figure 2: Illustration of how classification performance changes with codebook size increases.

6.2 When cluster size changes

As can be seen from Fig. 2, when the codebook size increases, the overall performance of both methods are decreasing. I think this is due to following reasons:

- The training set is too small, when we increases the codebook size, there is a chance the model will be overfitting. The histogram for each image would become sparse and it would be questionable to implement nearest neighbor on the sparse dimension as the distance is not well defined; For Bayesian method,the Gaussian distribution assumption along all clusters would also be broken, and we can see the effect of increasing codebook size is even higher than that of NN approach;
- The test set is also small, which will lead to a chance of higher variance. For example, if we look at the accuracy of Bayesian classification at codebook size 275. The highest can be over 0.95 and the lowest is even below 0.75. Therefore, more iterations may be needed to obtain a robust evaluation.
- After changing the codebook size, some implementation details should also be changed accordingly, which is not done in the exercise. For example, the cell size and number of cells of each patch should be adjusted. When we have a bigger codebook, a higher dimension of feature should be expected.

7 My own data

We also test the algorithms on other dataset, we use data from **https://www.csc.kth.se/cvap/actions/**. The original data are videos, we extrat the frames first and generate training and testing sets. Samples are shown in Fig. 3. We will classify the images into two categories: jumping and waving.

The classification are well done by both methods as illustrated by Fig. 4. This is as expected as the pictures are easy to recognize. One interesting observation is that although we use smaller size and roughly similar number of training sets, the overfitting doesn't happen. I think this is because compared to car, human takes up less space in the image and needs more detailed feature to recognize it. So increasing the cluster size won't lead to overfitting under the current range. But if we increase the cluster size to 500, the testing accuracy will drop to **0.8873** for NN and **0.9014** for Bayes. This is also in accordance with our analysis.

(a) Jumping (b) Waving

Figure 3: My own data: classification of human jumping and waving.

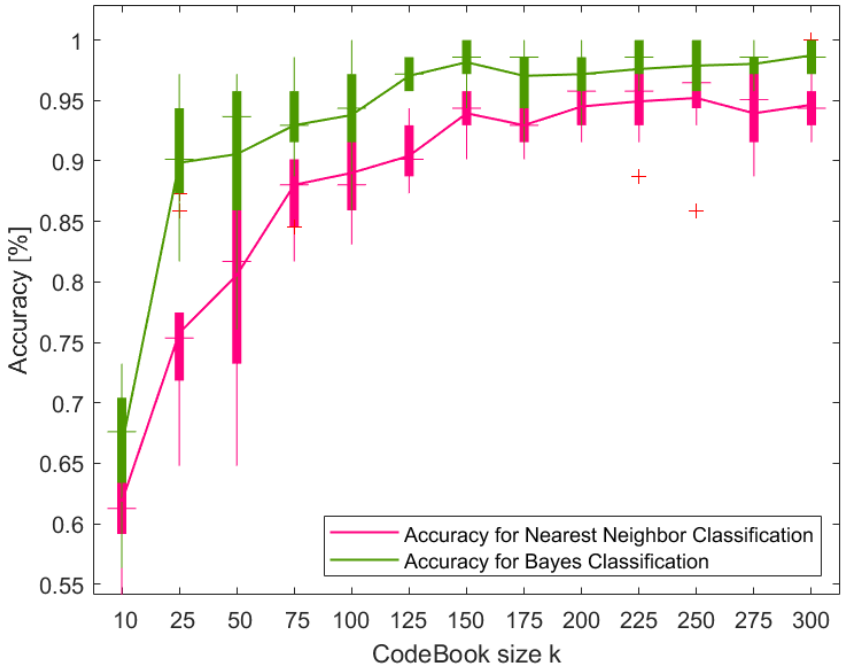

Figure 4: With my own data. the classification performance increases while the codebook size increases, and generally Bayesian outperforms NN approach.