Computer Vision

Assignment 10 : Specific object categorization

Autumn 2018

Nikita Rudin

December 18, 2018

In this lab we are going to implement an image categorization algorithm based on the bag of words technique. We will use a dataset of car images as training. We have 50 positive and 50 negative example images. We first extract features from positive train images and group them using k-means. These means become our words. We then construct a histogram of the occurence of these words for both positive and negative images. Finally we can extract features from train images, make the same histogram and then classify it as positive or negative. Two classifiers are compared: Nearest-Neighbour and Naive Bayes .

1 Feature extraction

We start by selecting patches from the image. We simply take patches around points on a predefined grid. We then compute the orientation of the gradients in this patch. Finally we subdivide the patch in 16 cells and compute a histogram of the oriented gradients in each cell. Using 8 bins we obtain a 128(16x8) feature vector. We extract approximately 100 of these features from every image. The MATLAB implementation is done the gradient() function, and histcounts() to create the histograms. After experimentation, it appeares that using atan() and not atan2() works better (setting the range of angles $+ - \pi/2$. This is makes sense because we are only interested in the orientation but not necessarily the direction of the gradients.

2 Codebook construction

Having extracted all features from the images containing cars, we can group them into k centres. We use k-means to achieve this goal. We initialize k centres to randomly selected features. We then compute the nearest center for each feature and and assign thus feature to that center. The centres are moved to the mean of the features assigned to them. We repeat these steps for a given number of iterations (10 in our case).
Finally, we obtain k features that represent the best our set of features. It is worth noting that this algorithm heavily depended on the initialization of the features. This is the main source of randomness in the classification. Together with the extracted features we saved image patches that produced them. An example of selected patches is shown below. We can see that a lot of these words represent something related to a car, but we also see some meaningless patches of constant colour.

Figure 1: Example of extracted codebook with k=200 (without the threshold trick).

3 Bag of words representation

We can now take an image. Extract the features and assign them to the closest word of the codebook. We then count the number of features assigned to each word and create an histogram across words for this image. This histogram is then compared to the histograms of positive and negative images for classification.

4 Nearest-Neighbour classification

This classifier simply computes the closest match for the histogram in the positiv and negative images. The two are then compared and the closest assigns its label to the image. This very simple approach gives very good results that are hard to beat by more complicated approaches.

5 Bayesian classification

The second classifier computes the probability that our image contains a car. In order to do this we the likelihood of getting a word when an image contains or not a car. We model this with a normal distrubution so we compute the mean and standard deviation of the counts of words in the histogram. We can then approximate the probability of getting the full histogram as the product of the probability of the words. This assumes that the words are independent, which is arguable.
If the probability of getting a given histogram is larger for the positive distribution, we classify the image as positive and vis-versa. For computaional reasons it is better to work in the log domain and sum the logs of likelihoods. It can happen that a certain word count has an std of 0 in our training set. We artificial set the std to minimal value of 0.5.

6 Results

The following results were obtained by averaging over 10 runs. Using k=200 words for the codebook we obtain the following results for the two classifiers.
Nearest-Neighbour : $accuracy = 92.29\%$ $std = 0.0337$
Bayesian classifier : $accuracy = 97.68\%$ $std = 0.0165$
It is important to note that the Bayesian classification worked poorly if we don't set a minimal std. It also works poorly if that minimal value is too high. As such it seems to work very well when it is properly tuned, but it is very much harder to get it to work.

Varying k doesn't seem to have an important impact on the results of the nearest neighbour classifier. For the Bayesian approach we see a clear increase of performance with increasing k. We can expect the Bayesian approach to perform worse for lower k, because we are simply giving it less information. The less words we have in the codebook the less distributions are used to calculate the likelihood.

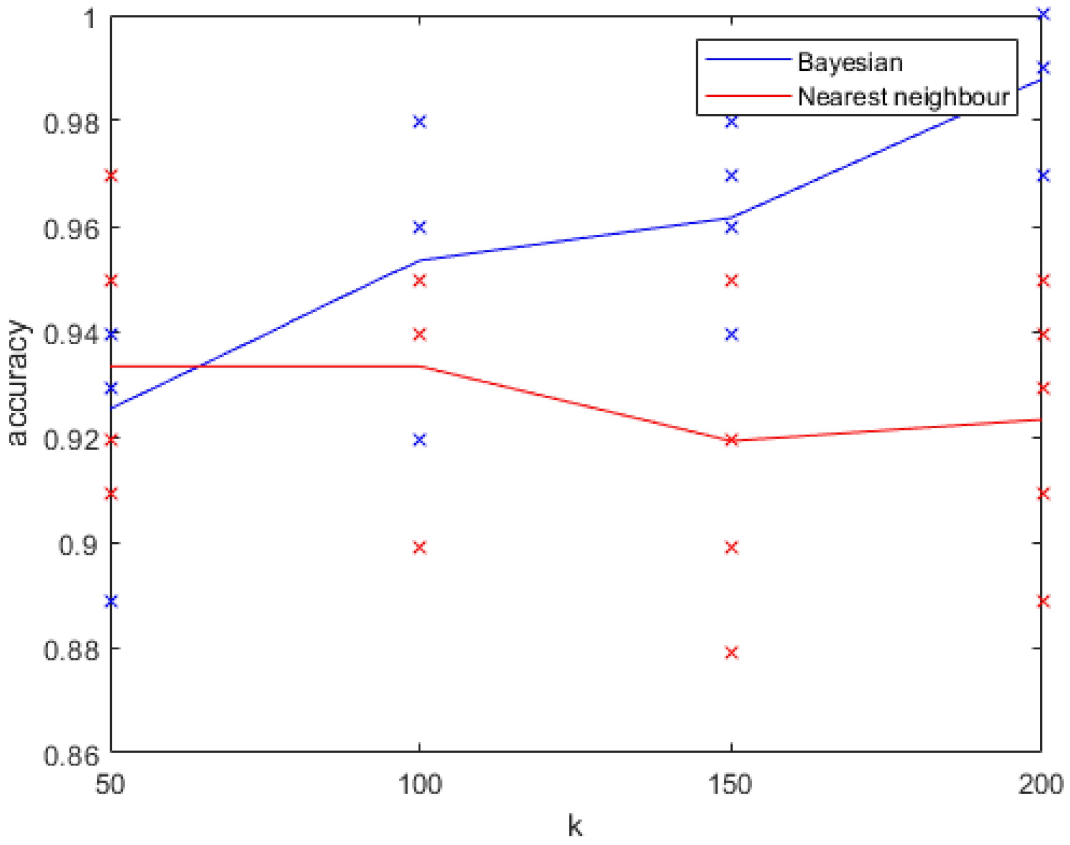

Figure 2: Effect of varying k from 50 to 200. Average over 5 runs for each value of k.