Computer Vision
Exercise 10: Image Categorization

Viviane Yang 16-944-530

11 December 2019

1 Local Feature Extraction

1.1 Grid Points

In order to compute the grid for a given image, we only need to use its dimensions. If we know the width and the height of a image, the grid can be computed by leaving out a border of 8 pixels around the image out and then choosing points on the new width and height every step size $\frac{width}{nPointsX}$ on the column axis and $\frac{height}{nPointsY}$ on the row axis.

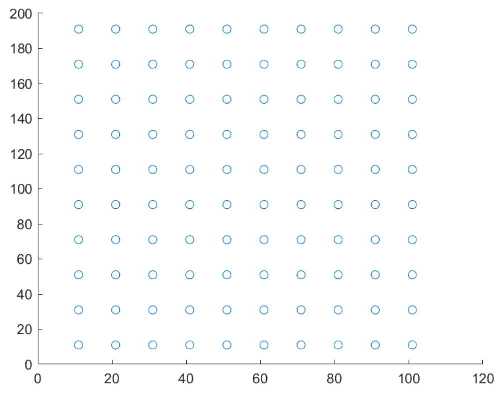

Figure 1: Grid points on a 120×220 matrix with a border of 10 and $nPointsX = nPointsY = 10$

1.2 Feature description

Using this grid points function, we can now compute the HOG descriptor of a image at these center points. We put a patch of size 16×16 pixels around a center point and divide it into $4 times 4$ cells. One cell has therefore a height and width of both 4 pixels. The orientation of the gradient is computed and for each cell a 8 bin histogram of the orientation is computed. The descriptor of a point is then the flattened out histogram of all cells in the patch. And the descriptors of a image is the set of all descriptors of each grid point. The MATLAB code looks like the following:

```
function [descriptors,patches] = descriptors_hog(img,vPoints,cellWidth,cellHeight)

    nBins = 8;
    nCellsW = 4; % number of cells, hard coded so that descriptor dimension is 128
    nCellsH = 4;

    w = cellWidth; % set cell dimensions
    h = cellHeight;

    pw = w*nCellsW; % patch dimensions
    ph = h*nCellsH; % patch dimensions

    descriptors = zeros(0,nBins*nCellsW*nCellsH); % one histogram for each of the 16 ...
        cells
    patches = zeros(0,pw*ph); % image patches stored in rows

    [grad_x,grad_y] = gradient(img);
    Gdir = (atan2(grad_y, grad_x));

    binEdges = -pi:(2*pi/nBins):pi;
    for i = [1:size(vPoints,1)] % for all local feature points
        center = vPoints(i, :); %2D point
        start_x = center(2) - pw/2;
        end_x = center(2) + pw/2 - 1;
        start_y = center(1) - ph/2;
        end_y = center(1) + ph/2 - 1;

        patch = Gdir(start_y:end_y , start_x:end_x);
        patch_img = img(start_y:end_y, start_x:end_x);
        patches(i,:) = patch_img(:);

        cells = mat2cell(patch, w*ones(1,nCellsW), h*ones(1,nCellsH));

        descriptor = [];
        for n = 1:nCellsW*nCellsH %16 cells
            cell = cells{n};
            hg = histcounts(cell(:), binEdges);
            descriptor = [descriptor hg];
        end
        descriptors(i, :) = descriptor;

    end % for all local feature points

end
```

2 Codebook construction

The codebook is consists of features of all pictures of the training set with positive and negative examples. We have the same number of pictures for the training and test set, therefore the dataset is not imbalanced. We can read the files of type **PNG** and **JPG** from two directories, one for the positive samples and one for the negative samples. Then, the grid points as well as the HOG features are computed for every picture, which together forms a dataset with high dimensional datapoints. By clustering these datapoints using k means, we can compute centers which belong to a category: car or no car. These center points form the codebook, whose patches can be seen in Figure [2].

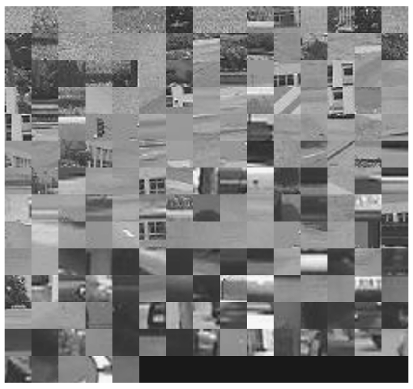

Figure 2: Cookbook of the car and no car images

3 Bag-of-words image representation

Now we want to compute the bag of words for each image. We compute how many features of the image belong to which centers and this is a indication for the classification, since similar images should have a similar distribution of features. Different histograms for training and testing set as well as one for positive and one for negative samples is created.

4 Nearest Neighbor Classification

For the nearest neighbor classification, we just look for the most similar histogram in the training set and take its label as the classification class. So if the histogram of the test image is closer to a histogram of the positive sample set than the minimal distance to the image in the negative sample set, we choose the positive class and vise versa.

5 Bayesian Classification

Using Bayes' theorem, we can compute the probability of a picture belonging to a class given a histogram can be also formulated with the probability of getting a histogram given that the image is a car, the probability of the image being a car and the probability of getting this histogram. The probability of getting a histogram given the class can be computed as the product of probabilities given the distribution of the class histograms. In order to do the computation in a numerical stable way, we take the logarithm of the probabilities and sum them up instead of multiplying them together. The normalization can also be left out, since we only want to know whether the probability of the image belonging or not belonging to the class given a histogram is bigger, their exact value is not the main point of interest. Based on these probabilities, we can do the binary classification.

6 Questions

6.1 What is your classification performance using the nearest neighbour classifier?

Evaluating over 10 runs, we get a classification accuracy of $p_{nearest} = 94.41$, which means there where 6 missclassification errors on average on the 99 test images.

6.2 What is your classification performance using the Bayesian classifier? Is it better or worse? How do you explain this difference?

The performance is a little bit worse, at an average accuracy of $p_{bayesian} = 93.43\%$for 10 runs. This could be since we are modeling the distribution of the bag of word features purely as gaussian, whereas the dataset is quite small. First of all, we don't know if gaussian is even the right model to choose and secondly we don't know if the dataset is big enough to find good parameters for the model. Nearest neighbor however just takes the class of the nearest neighbor and therefore works better on a small dataset.

6.3 Vary the number of cluster centres in your codebook, k. How does your categorization performance vary with k? Can you explain this behaviour?

Choosing a larger value than 200 for k does not make sense, since there are only 200 pictures in the training data sets and if we form a center for each image we can get 200 centers at maximum. By choosing a smaller k value, some of the centers lie in between the points, making the distance between centers and feature points larger, but also making the model more general and less prone to noisy features. Our dataset however is very clean, such that a decrease in k e.g. $k = 100$ leads to a decrease in accuracy, with $p_{bayes} = 94.95\%$ and $p_{nearest} = 91.92\%$. Now, the accuray of the bayes method is higher, since by assuming a gaussian distribution and taking the mean and variance, the model will not change too much with the number of k. However, with nearest neighbor, some centers disappear and get mapped to difference places, so the nearest neighbor might be from a point of the wrong class, since the center of the right class got shifted into a different direction.

7 Use your own dataset

I chose a dataset of flowers from the [1] dataset. In consists of 17 different flower categories with 80 pictures each. Since we are only doing binary classification, my first plan was to use the narcissus dataset as the class to detect and for the negative class I choose 80 random flower pictures. Then I splitted the data into 50 pictures each for training and the remaining 30 pictures each for testing. However, I got very bad accuracies at 55%, which is just a little bit better than random guessing. I explain that with first the general similarity of HOG features of plants. In the provided dataset of car and no car pictures, the cars are all photographed from the back whereas the no car pictures are picturing more or less empty streets. Their gradient difference is a lot bigger than the gradient difference between different kinds of flowers. Doing a classification between narcissus and the negative picture of the car data set (street picture) returns as expected a high accuracy of $p_{bayes} = 99\%$ and $p_{nearest} = 99.25\%$ over 10 runs. The application of differentiating different kinds of flowers is however more interesting, so I tested a non narcissus dataset consisting of purely picutures of a different kind of flower, orleanders. The accuracies improved for a bit, with $p_{bayes} = 75.67\%$ and $p_{nearest} = 73.17\%$ which is now a lot better than random guessing. For a multiclass classification problem, it is comparable of doing a one-vs-one and a one-vs-all voting approach. The one-vs-one approach is computationally more expensive, since we need to run the binary classification between each pair of classes. If there are n classes, we need to run $\frac{n(n-1)}{2}$ times for one-vs-one and only n times for one-vs-all. However, the accuracy can be higher than for one-vs-all.

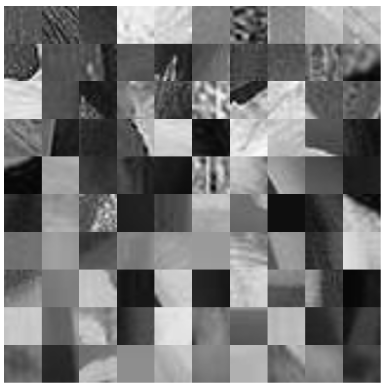

Figure 3: Flower BoW, using k=100

References

[1] Maria-Elena Nilsback and Andrew Zisserman. 17 category flower dataset. `http://www.robots.ox.ac.uk/~vgg/data/flowers/17/`.