Exercise for MA-INF 2213 Computer Vision SS22
13.05.2022
Submission deadline: 27.05.2022
Support Vector Machines (SVMs)

In their paper "Histograms of Oriented Gradients (HOG) for Human Detection" Dalal and Triggs [1] used HOG as feature vectors to train SVMs in order to detect people in images. The goal of this exercise is to understand their baseline algorithm and learn how to implement an SVM for object detection.
The algorithm itself works as follows:

- For each sample in a given set of training samples, select a detection window of a predefined size (the default size is 64×128), then extract a HOG descriptor for each detection window. The window specifies the region of interest from which you extract the HOG features.
- Use the extracted descriptors to train an SVM classifier
- Use the trained SVM to predict the class of the test samples.
- Vary the "hit threshold" to get different values of precision and recall

To solve the exercise, you may use the OpenCV implementation of SVM. A tutorial on how to use it can be found here: SVM tutorial.
We have provided you with the training and test datasets from Inria Person Dataset. In the following walk-through, we are going to assume that you are using OpenCV.

1. **OpenCV HOG**:
   In this task, you will use the OpenCV implementation of the HOG person detector to detect people in the given test images and annotate them with a bounding box. An example on how to use a HOG detector is provided on OpenCV website in a source file called "peopledetect.py". Note that you should use the already trained SVM in this task and evaluate over the test images which are located in $Sheet03/task\_1\_testImages$.
   (2 Points)

2. **Extracting HOG Features**:
   In this task, unlike in the previous one, you will extract the HOG features and train the SVM yourself based on the algorithm outlined above. Specifically, in order to train your own SVM, you have to extract HOG features from a set of positive and negative training images, i.e. images that contain the target object (person) and images that do not.

   – Extract HOG features from all the positive training images using HOGDescriptor::compute(). You need to crop each image to a size of 64 x 128. The training images are located in $Sheet03/task\_2\_3\_data$. In total, you will extract one HOG descriptor for every positive image.

- Now extract HOG features from the negative training images. Since these images are bigger (as described in [1] under "Methodology"), extract 10 patches (64 x 128) at random locations from each image, compute their HOG features and use them as your negative training samples.
- All training features together with their corresponding label information should be stored and then passed to the SVM object for training (See Task 3).

(4 Points)

3. **Train the SVM**:

   In case of a linear SVM, the training algorithm finds the hyperplane which best separates the positive training samples from the negative ones. In many cases, the data contains outliers or is not linearly separable, and then you allow the classifier to misclassify some samples for a cost such that you can increase the margin. The cost of this error is determined by how far the sample is from the margin. The value of the parameter C which you have to choose during training determines the trade-off of reducing the margin versus the number of misclassifcations.

   Let us now examine the influence of the value C on the results.

   - Choose $C = 0.01, C = 1$ and $C = 100$, i.e. train three different SMVs (using the HOG features extracted in task 2).
   - Use all 3 SVMs to classify the given test images. For the positive images, there is only one HOG desciptor per image. For the negative images, there are multiple HOG descriptors, depending on the size of the input image. Make sure your patches have dimension 3780.
   - Create a file containing the confidence scores (distance to the margin) for each training sample. What do you observe when you visualise the results for different values of C? *(3 Points)*

4. **Plotting the results**:
   In general, there is a trade-off between precision and recall, since increasing one usually happens at the cost of the other. In this task, you are going to examine the influence of the choice of C by presenting the results as precision-recall plots. In this plot, you vary the hit-threshold which gives a set of (recall, precision) value pairs representing the (x, y) values on the graph.

   The formulas for these two metrics are given by:

$$precision = \frac{TP}{FP + TP} \qquad recall = \frac{TP}{FN + TP},$$

where TP = true positives, FP = false positives, FN = false negatives. You will get different classification results from the SVM by changing the threshold. For plotting the results, you are free to use any tool that you are familiar with: Matplotlib in Python, GNU R or Matlab. *(3 Points)*

5. **Multiple Detections**:
   While applying your trained SVM to the HOG descriptors that you extracted from an image, you have encountered the problem of duplicate detections. This means that one person in an image is detected in multiple windows and scales and naturally you would like to pick only the best one. To get the correct number of detections, one has to eliminate some of these detections:

   - First write an algorithm that slides a detection window across the image. Use a scale factor $s = 1.2$ for multi-scale detection, that is, you should apply a scale factor of 1.2 to create a pyramid of downscaled images until the size of the image becomes equal to the detection window, then slide your detection window across the image for each scale. Feel free to experiment with different values for s in the range $[1, 1.5]$. to examine its influence
   - Use your trained SVM with a specific threshold (see task 2) to decide which windows are classified as positive detections.
   - Use non-maximum suppression to eliminate redundant positive windows based on their overlap area and confidence scores.
   - Apply your algorithm to the same test images as in task 1 using the SVMs trained in task 3 and visualize the detections before and after applying non-maximum suppression (NMS), then save the classification results into a file. What is a good value for the elimination threshold when applying NMS?
     Bonus (1 point): In NMS, the naive solution requires $O(n^2)$ time complexity where n is the number of detections, can you think of a solution to reduce it to $O(n \log(n))$.

   (8 Points)

References

[1] Dalal N. and Triggs B., *Histograms of Oriented Gradients for Human Detection*. CVPR 2005