Exercise for MA-INF 2213 Computer Vision SS18
18.05.2018
Submission deadline: 06.06.2018
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 image classification.
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 (1 indicates a person and -1 indicates not a person)
- 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 on the web [2].
We have provided you with the training and test datasets. In the following walk-through, we are going to assume that you are using OpenCV.

1. **OpenCV HOG**:
   In this section, 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.cpp". Use the already trained SVM. The test images are located in $Sheet04/section\_1\_testImages$. *(4 Points)*

2. **Extracting HOG Features**:
   In this section, you will extract the HOG features and train the SVM yourself. 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 $Sheet04/section\_2-3\_data$. 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 (positive label = 1.0 , negative label = -1.0) should be stored and then passed to the SVM object for training (See Section 3).

(3 Points)

3. **Train an 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 is not linearly separable and then you allow the classifier to misclassify some samples for a cost. 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 section 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 an Ascii file containing the confidence scores (distance to the margin) for each training sample. These scores can be obtained by CvSVM::predict(...,returnDFVal=true). What do you observe when you visualise the results for different values of C?

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 section, you are going to examine the influence of the choice of C by presenting the results as precision-recall plots. *(5 Points)*

   The formulas for these two metrics are given by:

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

   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 use any tool that you are familiar with, such as Matlab, or some other free plotting tools: Matplotlib in Python,QTI-plot, GNU R. For GNU R there will be plotting script online which you can use and adapt. GNU R is available for Windows and Linux ("R Commander" in the Ubuntu Software Center).
(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 detection. This means that one person in an image is detected in multiple windows and scales and naturally you would like to pick the best one only. 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 of 1.2 for multi-scale detection.

   – Then use your SVM with a specific threshold (see section 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 section 1 using the SVMs trained in section 3 and then save the classification results into a file.

   (7 Points)

References

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

[2] *http://docs.opencv.org/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html*