CSE527 Homework 4
Real-time object detection using multiple resolutions

Due Thu, Nov 20, 2014

1. Description

In this homework, you will implement a multi-resolution, multi-scale framework for detecting pedestrians. The details for the algorithm can be found in this paper, which is also included in the homework file:

W. Zhang, G. Zelinsky, and D. Samaras. Real-time Accurate Object Detection using Multiple Resolutions. In ICCV07.
http://www.cs.sunysb.edu/~ial/content/papers/2007/wzhang-iccv2007.pdf

You will have a set of positive images (with pedestrians) and negative images (without pedestrians) for this homework. Each image is downsampled to create images of multiple resolutions. In each image, HOG features are calculated for multiple window locations. Using these features, Support Vector Machine (SVM) classifiers for detecting pedestrians are trained for each resolution, and these classifiers are used in the test images to find pedestrians.

2. Tools

You can use existing code for SVM classification and HOG extraction. A good library for training and testing SVMs is LIBSVM. The instructions for using LIBSVM in Matlab can be found in the 'matlab' folder in the LIBSVM package. Also, here are some useful links if you decide to use this library:

LIBSVM website: http://www.csie.ntu.edu.tw/~cjlin/libsvm/
LIBSVM documentation: http://www.csie.ntu.edu.tw/~cjlin/papers/libsvm.pdf
A practical guide to SVMs in general:
http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf

You can use VLFeat or other computer vision libraries for extracting HOG features.

VLFeat website: http://www.vlfeat.org/
VLFeat HOG tutorial: http://www.vlfeat.org/overview/hog.html

3. Dataset

For this homework, you will use the INRIA person dataset (*Dalal and Triggs. Histograms of Oriented Gradients for Human Detection. In CVPR05*). The dataset consists of positive images (with people) and negative images (without people), along with corresponding annotation files. It also contains positive images cropped in 64x128 pixel format. You can download the dataset from:

http://pascal.inrialpes.fr/data/human/

4. Algorithm Steps

The paper describes the flow of the training and detection algorithms in detail. Here is an abridged version for your convenience:

4-1. Initial setup

a. Choose your downsampling ratio for scale and resolution. Let us denote the resolution ratio as α, and the scale ratio as β. For example, if you want the resolution to be halved at each level, then α = 2. Based on these ratios, define your resolution & scale space. The resolution will be r = 1, … , R, where 1 represents the lowest resolution and R is the highest resolution (which is the original image resolution). Similarly, the scale will be s = 1, … , S, where 1 represents the largest object and S is the smallest.

b. Decide your initial object window size (w,h). Since the INRIA dataset already contains cropped image patches of size 64x128, it would be a good idea to use the same size for your window.

4-2. Training (40 points)

a. (10 points) Create a set of positive image patches (which contain people) and negative image patches (which don't) from the positive and negative training images. Use the window size you defined in step b. These original patches belong to the full resolution level (r = R).

b. (10 points) Downsample the full resolution image patches to your lowest resolution. In other words, downsample them with a ratio of α^{R-1}. Then extract HOG features from all patches. Use the block-based HOG feature extraction that is described in the paper (section 2.3).

c. (10 points) Train an SVM classifier for HOG features of the lowest resolution. Label the features from the positive patches as positive instances, and those from the negative patches as negative instances. We'll denote this classifier as C_R.

(A tip for LIBSVM users: when you use svmtrain() function, the columns of your training data matrix should be the features, and the rows should be the training instances. For example, if your feature dimension is 128 and you have 200 training instances, your matrix should be 200x128 in Matlab.)

d. (10 points) Repeat steps b and c for each resolution r = 2, … , R. Downsample the original positive and negative patches with a ratio of α^{R-r}, and train an SVM classifier C_r for HOG features in resolution r. In the end you should have R separate classifiers for each resolution.

4-2. Detection (30 points)

a. (10 points) Start with the lowest resolution (r = 1) by reducing your window size by a factor of α^{R-1}. For each scale s = 1, … , S, downsample the original image with a ratio of $\alpha^{R-1}\beta^{S-1}$, but keep the window size fixed. By doing this, you are keeping the feature resolution but changing the object size in effect. Apply the classifier to each possible window

position in the image. You should choose how much the window will move across the image each time. Mark each window with its classification results (positive or negative).

b. (10 points) Go to the next lowest resolution (r = 2). Your detection window should now be $1/\alpha^{R-2}$ of the original size. Again, for each scale s = 1, … , S, downsample the image accordingly. For each scale, apply the classifier to windows that were marked as positive in the previous resolution. Skip all windows that were marked as negative or were ignored.

c. (10 points) Repeat above steps for all other resolutions (r = 3, … , R).

4-3. Evaluation (30 points)

Run pedestrian detection tests using the INRIA training and testing image set. You should keep your training and testing set balanced by having roughly the same number of positive and negative examples. After you run the test, calculate your detection accuracy by looking at how many windows detected people correctly or incorrectly. To do this, compare the window locations with the ground truth bounding boxes that are provided in the INRIA dataset. Also plot the receiver operating characteristic (ROC) curve for false positive and true positive rate. If you're using LIBSVM, the plotroc() function can help you with the task. (plotroc.m is included in the homework file.) Try running the test with different parameters (resolution and scale ratio, etc.)

5. What to Submit

Submit a report, and your code for the whole framework.
In the report, you should include detailed explanation about the evaluation. Include examples of your positive and negative image patches, and write down the exact size of your training and testing image set. Report your detection rate, along with the ROC curve and the parameters that you used (number of resolution and scale levels, downsampling ratio, detector window and HOG parameters for each resolution, etc.) Have a look at the paper's results section and try to understand how the authors did their evaluation.

6. Extra Credit Problems (60 points)

a. (10 points) Implement your own version of the HOG extraction function so that it can control the amount of stride for blocks and the number of cells in each block. In the evaluation phase, try changing these parameters to find the best result.

b. (30 points) It is known that acquiring hard negative samples from the classifier and re-training it using those samples can significantly improve performance. "Hard negative samples" mean the negative training instances which were incorrectly classified as positive during the training phase. While training the SVMs in each resolution level, you can run the SVM on negative image patches, pick the ones that are marked as positives (which are, in fact, false positives), and then include them in the negative training set and train the SVM again. Implement this re-training process in your framework. Run the test again, and compare the results between the algorithm with and without re-training.

c. (20 points) The paper applies the detection framework to objects other than people. Download the Pascal Visual Object Classes (VOB) challenge 2006 database (http://pascallin.ecs.soton.ac.uk/challenges/VOC/voc2006/index.html). Use your code to run detection tests on two other object classes. Calculate the accuracy and ROC curve for each object class, and compare the result with the people detection problem. If the detection rate is different between object classes, can you figure out why? Write a brief explanation about why one object class might be more easily detected than the other.