Boosting Laboratory

Tae-Kyun Kim
Imperial College London
June 2014

In this lab session, we learn the machine learning algorithm called Boosting by Matlab. Following the lecture on Boosting, we implement the algorithm, see through the core parts of the algorithm step-by-step, while training and testing it on the CMU Face Detection data set. Try also different parameter values of Boosting and see their effects. Although the lab session is limited due to the given time and efficiency of the Matlab codes provided, for a real-scale problem, ponder about why Boosting is so important in computer vision and machine learning and in comparison to other alternative methods, in terms of scalability, efficiency, multi-classes, and generalisation. For its various up-to-date application work, please have a look at http://www.iis.ee.ic.ac.uk/ComputerVision.

Instructions to use the provided codes and data for object detection: First install a C compiler if you do not have one in your computer. Any C compiler might work. In case that you install one, you are recommended to use MinGW. http://www.wikihow.com/Install-MinGW-(Minimalist-Gnu-C/C%2B%2B-Compiler)-on-Microsoft-Windows. This link shows a very detailed tutorial on how to install MinGW on Windows.

Download and unzip Boosting.zip.

Face Detection and Boosting

Open and refer to **script.m** in the root directory of the project. It provides the whole code that you need to run.

Run **setup.m** to compile all c files. The compiler options may pop up. Choose your default C compiler. Once they are successfully compiled, they produce xxx.mexw64 files or the similar.

1. (Training data collection) Load **ImgData_tr.mat**, in which face images (they are already cropped and resized to 24x24 pixels) are stored in **ImgData_tr.Pos**, and random square image patches for non-face data are in **ImgData_tr.Neg**. (*We need to collect as many random patches as reasonable from *various source images*, and to resize them to 24x24 pixels. Let us say 10 times more non-face image patches than the face image patches provided.) Store them in **ImgData.Pos** and **ImgData.Neg** respectively.

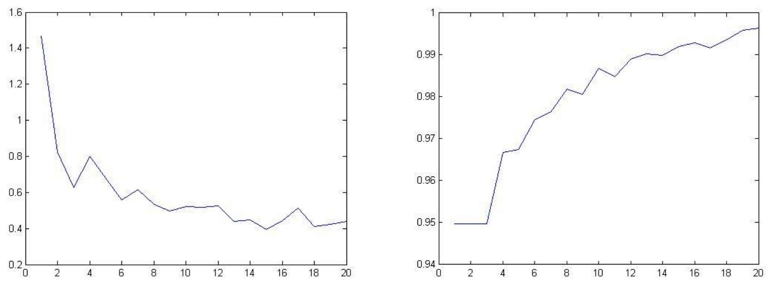
   For the purpose of training here, never use the testing data (ImgData_te.mat) and the 5 test images.

2. (Adaboost learning)

   - Run **DataProcess** and **WeakLearnerList**, to get all data in the necessary format and to build the list of weaklearners. See into the codes, if necessary.

- Study the lines of code for the **Adaboost_Harr** function.

- Learn the Adaboost classifier. Show the alpha values and the recognition accuracies i.e. correct classification rates on the training data at different boosting rounds. Below is the example result.

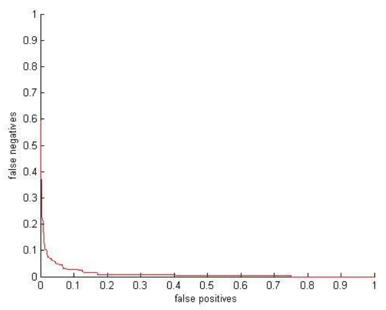


Alpha values (left) and training recognition accuracies (right) through the boosting rounds

3. (Evaluation: recognition accuracy %) Evaluate the learnt boosting classifier on the provided test data set **ImgData_te.mat**, which contains both face and non-face image patches in the size of 24x24 pixels.

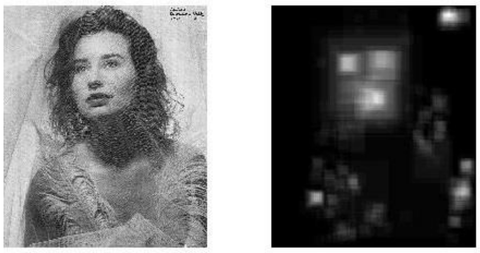
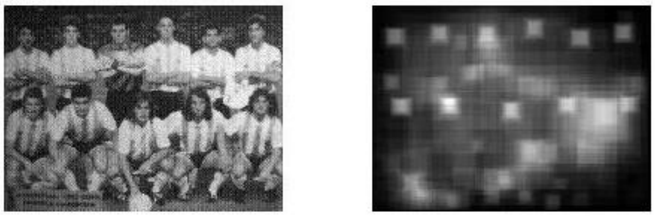
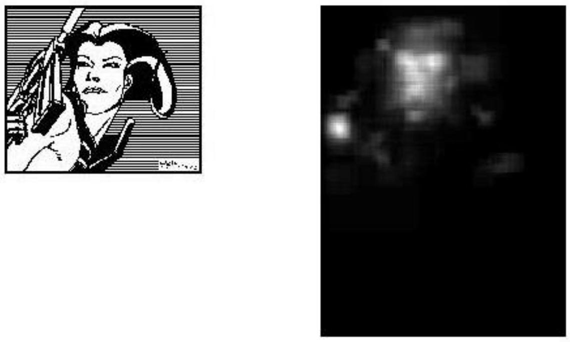
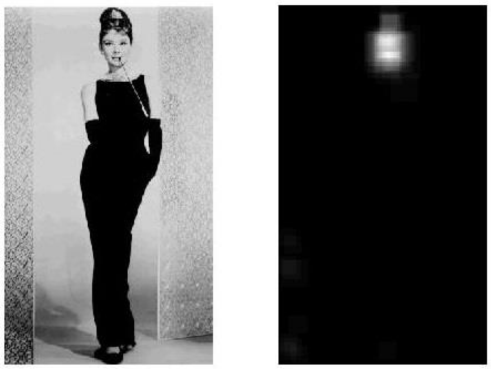
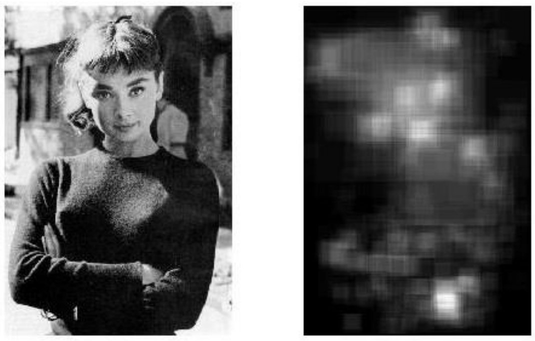
- Calculate the recognition accuracy i.e. correct classification rate (%) on the test data. Aim at achieving the accuracy around 97 %. (You can use the part of the code **Adaboost_Harr** to compute the recognition accuracy)

- Draw the ROC curve (you can use the **roc** function in the statistical pattern recognition toolbox or write your own codes). Aim at achieving the similar roc curve below.



Instructions to install the statistical pattern recognition toolbox: download it from http://cmp.felk.cvut.cz/cmp/software/stprtool/, and follow the detailed instructions written in readme.txt in the root directory of the toolbox. Set the path by typing **stprpath** in the Matlab command window.

4. (Evaluation: face detection in images) Use the provided test images in the directory of **./test_images/**. Get the inputs in the necessary format for the function, **findface**. See into the code, if necessary. Show the response maps. Below is the example result.

5. [Additional] Try to improve the boosting classifier accuracy, 1) in terms of the recognition accuracy (%) and ROC curve on the provided test data (**ImgData_te.mat**) and/or 2) in terms of the quality of the response maps on the provided test images. Explain about the ways you tried or would have tried. You can use the images in the directory of **CMU_FD_DB,** and data in **BancaData.mat** and **MPEGData.mat**.

In case of your further interest, refer to Viola and Jones, robust real-time object detector, CVPR01 (http://www.iis.ee.ic.ac.uk/icvl/mlcv/viola_cvpr01.pdf).