EE462, EE9SO25, EE9CS728

Machine Learning for Computer Vision

Hands-On Session for Adaboost for Face Detection

Tae-Kyun Kim
Imperial College London
Jan 2018

In this lab session, we learn the machine learning algorithm called Boosting by Matlab. Following the lecture on Boosting, we implement the algorithm, see the core parts of the algorithm, 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, discuss why Boosting is popular for a real-time and scale problem in the field of computer vision and machine learning. How do we compare it with alternative methods e.g. randomised forests, in terms of scalability, efficiency, multi-classes, and generalisation. For some variants/applications, you may have a look at https://labicvl.github.io/.

Instructions for the provided codes and data for object detection (not needed in lab computers): You need to configure Matlab to compile .mex files. First install a C compiler if you do not have one in your computer (e.g. MinGW in Windows, Xcode in Mac, gcc in Linux). 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.

Download and extract 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.

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.

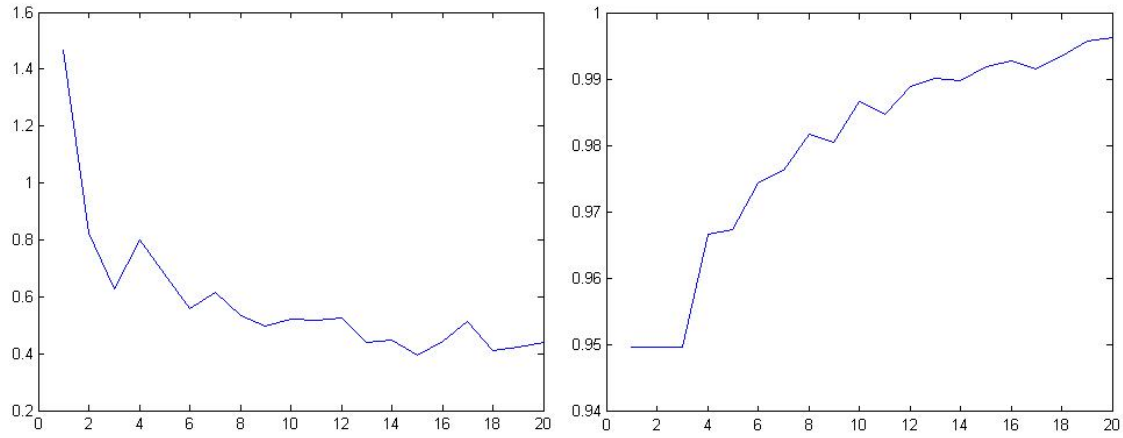
In the training phase, do not use the testing data (ImgData_te.mat) and the 5 test images.

2. (Adaboost learning)

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

- 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 of the training data at different boosting rounds. Below is the example result.

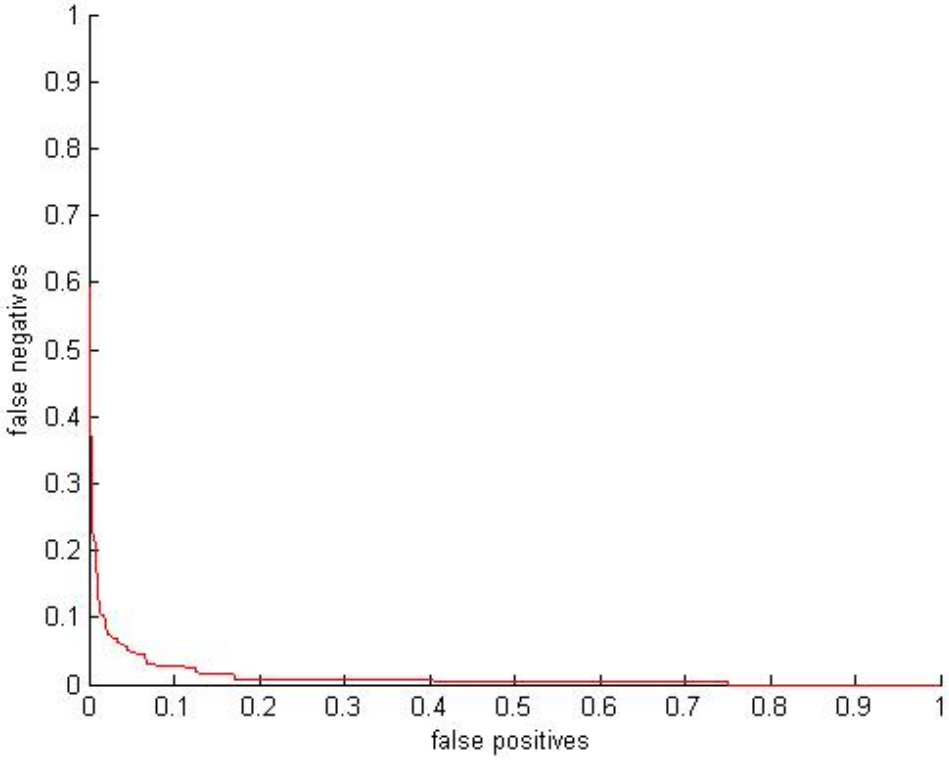


Alpha values (left) and training recognition accuracies (right) through the boosting rounds

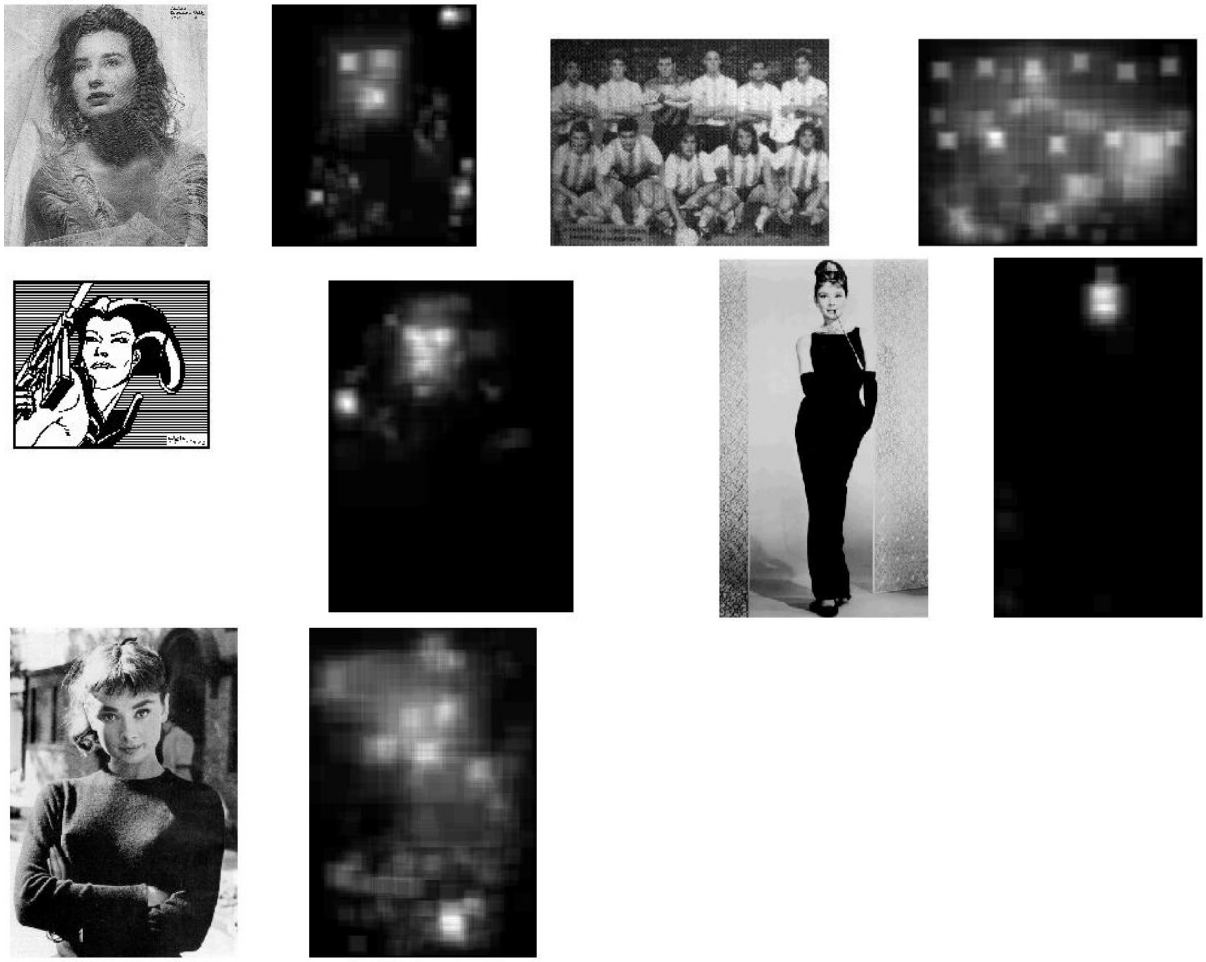
3. (Evaluation: recognition accuracy % of the testing data) Evaluate the learnt boosting classifier on the provided test data set **ImgData_te.mat**, which contains both face and non-face image patches unseen in the training stage, in the size of 24x24 pixels.

- Calculate the recognition accuracy i.e. correct classification rate (%) of 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** provided or write your own codes). Aim at achieving the similar roc curve below.



4. (Evaluation: face detection in the test 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 needed. Show the response maps. Below are example results.



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 i.e. brighter and peakier responses around faces and less false alarms. Explain the ways you improve. You can use the images in the directory of **CMU_FD_DB**, and data in **BancaData.mat** and **MPEGData.mat**.

In case of your interests, please refer to Viola and Jones, robust real-time object detector, CVPR01 (http://www.hpl.hp.com/techreports/Compaq-DEC/CRL-2001-1.pdf).