VIOLA-JONES FACES RECOGNITION

ALESSANDRO MANZOTTI

In this homework, we consider the problem of face detection. That is, given an image, we wish to identify and locate instances of a general object class, i.e. faces within the image. A complete object detection system consists of a sliding window and a classifier. As the sliding window scans the image, the classifier makes a decision regarding whether or not the portion of the image occupied by the sliding window is an instance of a particular class or not. Obviously, such a classifier must have a very low false positive rate as the number of negative samples will drastically outnumber the positive samples. We implement an Adaboost classifier with Haar-like features. Such a classifier consists of a weighted sum of many weak classifiers. To obtain a sufficiently low false positive rate, we cascade several Adaboost classifiers, where at each stage of the cascade only the samples classified as positive are allowed to pass through. Thus, increasingly difficult false positives are pruned away at each stage.

1. Haar-like features

The training images considered in this homework are of size 64×64 pixels. From these image, we extract a subset of Haar-like features, which are computed using vertical and horizontal differencing operators. The features are computed efficiently exploiting the integral image. Given the computational power and time I decided to use just the simple rectangular Haar feature starting with a minimum size of 6 pixels (3+3 pixels). I use a stride of size 4 in the final run. In total I have used a subset of **61440 features**.

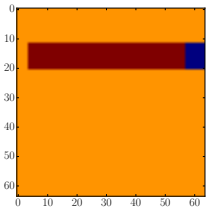

Figure 1. An example of an important feature (weak classifier for AdaBoost) for face-classification.

2. Single AdaBoost

As suggested I have used a simple AdaBoost. The final classifier threshold was set to avoid any false negative (depending on the stage it was $1.3 < \Theta < 3.2$). An example of one of the best weak classifier and the train error for a single AdaBoost are shown in Fig. 1-2-3.

3. Cascade

Being computationally limited we fixed the final cascade number to 6 AdaBoost classifiers. Training time and errors are showed in Table 1.

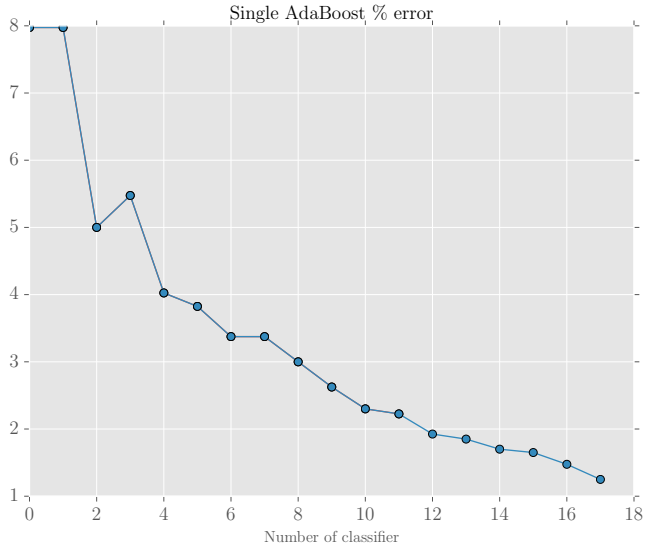

FIGURE 2. Training error for a single AdaBoost as a function of the number of "weak lerner" used.

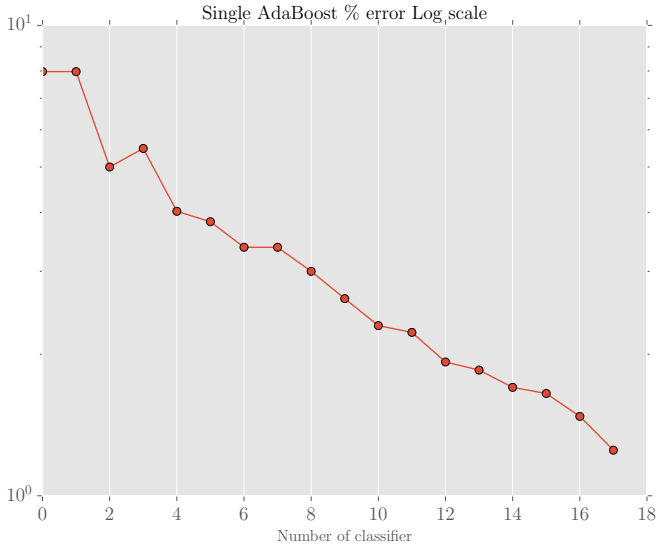

FIGURE 3. LOG SCALE. Training error for a single AdaBoost as a function of the number of "weak lerner" used. As can be seen and as expected the error decrease exponetially.

4. Evaluate on images

I just use a sliding window on the test image. To save time I did not check all the possible patches but I move the sliding window for 8 pixels at each move (either vertical or horizontal). This still takes a lot of time and a parallelization of the feature computation and patch evaluation would probably be needed for this.

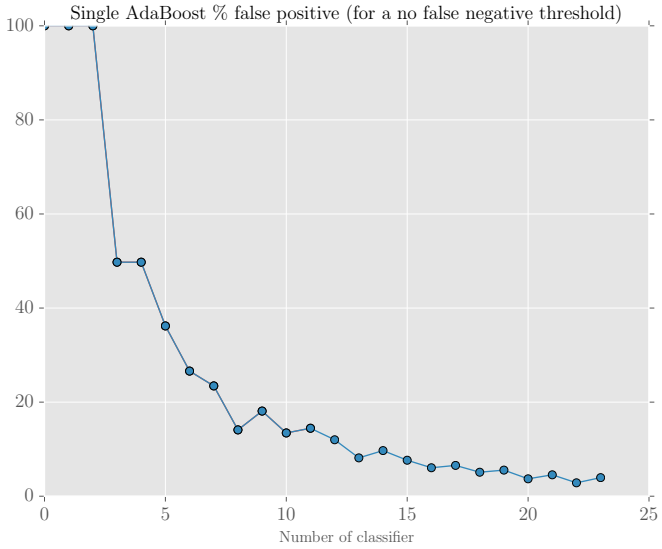

FIGURE 4. This shows the number of false positive for a single AdaBoost (no cascade) as a function of the number of "weak lerner", Haar features in this case, used in the classification. In this case a threshold was set so that there were no false negative. With 5 features we can push the error to 18%. Using a cascade of classifier we can achieve a fairly low rate of false positive.

TABLE 1. TRAINING THE CASCADE. These refers to 61440 features and no parallelization implemented in the Python code. The classifier threshold was set to avoid false negative.

Stage	Training time (s)	Error
1	220	10%
2	390	8%
3	390	6%
4	421	3.7%
5	513	2.8%
6	636	2.5%

5. Overlapping and plot

I noticed that most faces are detected at multiple nearby positions, while false detections often occur with less consistency. I tried to use this observation to eliminate some false detections. For each location at which a face is detected, the number of detections within a specified neighborhood of that location can be counted. If the number is above a threshold, then that location is classified as a face. For this reason I applied a Gaussian filter to the binary matrix (1 face 0 non face). After that I applied a threshold to retain only a small percentage. The goal of this was to reduce the overlapping while excluding very isolated, and for this reason likely false, positive detection.

After that I finally reduce the overlapping with a simple non-maximum suppression edge thinning technique. Basically I computed the area overlapping of all the squared 64x64 detection patches and discard all of those that overlaps for more than a threshold value.

6. Results

The final results is shown in Fig.5. Given the fact that I have used given time and computational power, not many and very simple Haar features, the classifier picks all the patches with a significant dynamic contrast. This means not only faces but also hands and patterns in people dresses. Significant is the example of the girl in the lower-right corner. Overall however a fair number of faces has been detected.

Figure 5.

7. Possible improvements

- More Haar features. To implement that, a simple parallel code might be needed. Most of the training and evaluation can be embarassily parallelizwd among the 4000 images (training) or patches (evaluation). This would have probably allowed me to use more features, while keeping the code in a reasonable time frame. The main issues however was the size of the features. I tried to save them as float16 but still it was a significant challenge for my laptop.
- Embarassing parallelism.
- Use margins. Indeed one can use the last Adaboost classifier of the cascade to give also an estimate of the significance of each detection. Notice that this margin, $|\sum_{t=1}^{T} \alpha_t h_t(x)|$, is what AdaBoost tends to maximize. This information can be passed to the gaussian filter (for example not using a binary matrix detected-not detected but a matrix of the significance of the detection) and the overlapping filter to make a more informed guess.