Face Detection using Viola Jones Algorithm

Shruthi Sampathkumar

December 9, 2019

1 Abstract

Face Detection algorithm of Viola Jones introduces concepts of feature extraction that makes the algorithm unique.

2 Feature Extraction

: The feature extraction in Viola Jones is done using the following concepts

- Finding Integral Image Calculation of rectangle region values from the integral image.
- Finding features that consists of regions positively contributing to the feature and the regions that are negatively contributing to the feature.

2.1 Integral Image:

Integral Image of any image has the shape of the original image. It is calculated for every pixel of an image. The idea is that:

$$integral\_image[i][j] = \sum_{x=0,y=0}^{i,j} image[x][y]$$

Thus the integral image would have the same shape as the original image. This functionality is implemented in the function *integral_ image()* in the code.

2.2 Pixels in rectangular regions:

The integral image can now be used to find values for each of the rectangles that could formed in the image. This value represents the summation of all the pixel values in the actual image that is contained in a particular rectangle in the image. Thus, this concept could be used to know how much a particular region in the image is **darker/lighter** than the other regions which could be further used to **differentiate the edges, curves, shades** of a face. This is implemented using the *calculate()* method in the code.

2.3 Positive and Negative Rectangles:

Each of the rectangles that was found in 2.2 is classified as positive or negative region as proposed by Viola Jones the paper –. These are the features that will be used in our Adaboost algorithm to classify the images and are called as Haar Features. They are in the form as shown in Fig.2.1. The regions in white rectangle contributes negatively to

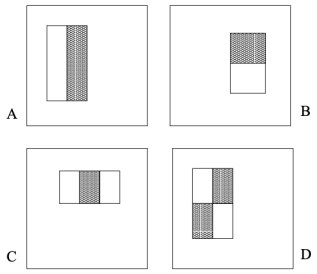

Figure 2.1: Division of positive and negative regions of an image

the feature value and the ones in shaded contributes positively to the feature value.

A and B in Fig.2.1 as two rectangle regions, the first one being horizontal and second one being vertical while C is a three vertical rectangle region type and D represents a four rectangle region type. The width and height of the rectangles are at max 8 in our implementation to reduce the number of features that could be generated. Thus the number of Haar features generated by our implementation is very much lesser when compared to the number of Haar features that could be generated for an image.

Thus features are extracted for each image. We have 5 types of features for the images in our dataset : two types of two rectangle regions (one horizontal and one vertical), two types of three rectangle regions (one horizontal and one vertical) and one type of four rectangle regions. The specific number of regions for an image of size 19x19 and the filter for width and height of the rectangles as 8 is as discussed below.
The total number of (Haar) features is **46656**.
Number of Horizontal two rectangle regions (Type 1) = 11340.
Number of Vertical two rectangle regions (Type 2) = 11340.
Number of Horizontal three rectangle regions (Type 3) = 7938.
Number of Vertical three rectangle regions (Type 4) = 7938.
Number of four rectangle regions (Type 5) = 8100.

3 Training:

The features extracted using the techniques that are discussed in section 2 is used for training the Adaboost model for several rounds and the performance changes are observed. The goal is to choose the best feature for every round of the Adaboost classifier. This is done by iterating over each of the extracted features f_i where $i = 1, .., m$ ($m = 46656$ in our experiment) of the input and iterating through all samples for each of the features f_{ij} where $j = 1, .., n$ ($n = 2500$ in our experiment). In round k, the best classifier is found by using the empirical error of each of the features. Parity and threshold for each of the features is also found by using the false positive and false negative rate.

```
Round 1/10
Feature type :  Two Horizontal
Feature (left,top,right,bottom) :  ([[8, 2, 1, 8]], [[7, 2, 1, 8]])
Threshold :  105.0


Performance Evaluation
Accuracy :  0.7844723008491711
False Positive Rate :  0.09195402298850575
False Negative Rate :  0.739406779661017
```

Figure 3.1: Training results of the model after Round 1

```
Round 3/10
Feature type :  Four Rectangle
Feature (left,top,right,bottom) :  ([[7, 14, 3, 2], [4, 16, 3, 2]], [[4, 14, 3, 2], [7, 16, 3, 2]])
Threshold :  39.0


Performance Evaluation
Accuracy :  0.7808329963606955
False Positive Rate :  0.06096951524237881
False Negative Rate :  0.8898305084745762
```

Figure 3.2: Training results of the model after Round 3

The best feature and the corresponding threshold and parity that are thus found for each round are stored in their respective lists.

4 Evaluation:

The evaluation of the model is done by using the decision stumps that were produced by training. The features that were chosen are applied to the test input sequentially and a cumulative value is used for it's final classification. This is described by the formula in

$$prediction(x) = alpha_t * 1(polarity_t * feature_{tx} < polarity_t * threshold_t), t = 1, .., T$$

```
Round 5/10
Feature type :  Two Horizontal
Feature (left,top,right,bottom) :  ([[12, 4, 3, 6]], [[9, 4, 3, 6]])
Threshold :  647.0


Performance Evaluation
Accuracy :  0.7856854023453296
False Positive Rate :  0.04697651174412794
False Negative Rate :  0.923728813559322
```

Figure 3.3: Training results of the model after Round 5

```
Round 10/10
Feature type :  Two Horizontal
Feature (left,top,right,bottom) :  ([[12, 4, 3, 1]], [[9, 4, 3, 1]])
Threshold :  14.0


Performance Evaluation
Accuracy :  0.7731500202183583
False Positive Rate :  0.06346826586706647
False Negative Rate :  0.9194915254237288
```

Figure 3.4: Training results of the model after Round 10

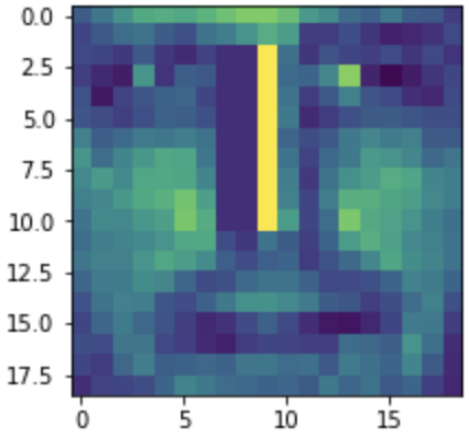

Figure 5.1: Top feature chosen by round 1

where $prediction(x)$ is the classification result for the test image x, T is the number of rounds, $feature_{tx}$, $alpha_t$, $polarity_t$, $threshold_t$ are the values of sample x computed in the t round using the classifier chosen for the round t.

If

$$prediction >= sum(aplhas),$$

then classification is 1 else classification is 0.

5 Results:

The results of each of the training rounds 1,3,5 and 10 is shown in figures 3.1, 3.2, 3.3 and 3.4 respectively.

The top feature chosen in each of the rounds 1,3, 5 and 10 are shown in figures 5.1, 5.2, 5.3 and 5.4 respectively.

The evaluation results of the model for each of the rounds 1, 3, 5 and 10 on the test data is shown in the Figure 5.5

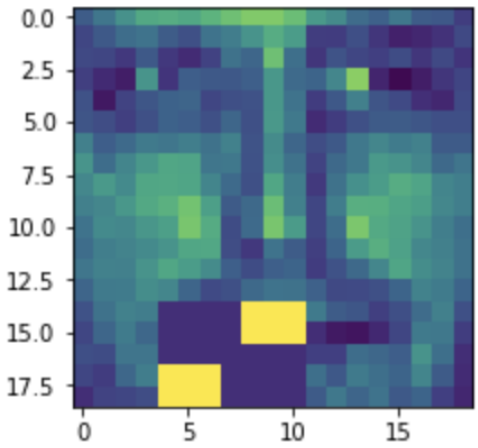

Figure 5.2: Top feature chosen by round 3

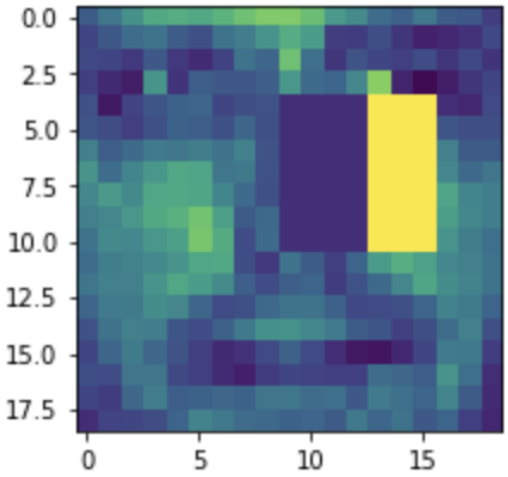

Figure 5.3: Top feature chosen by round 5

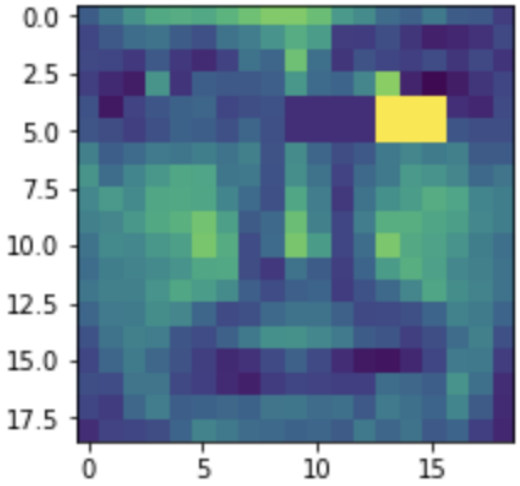

Figure 5.4: Top feature chosen by round 10

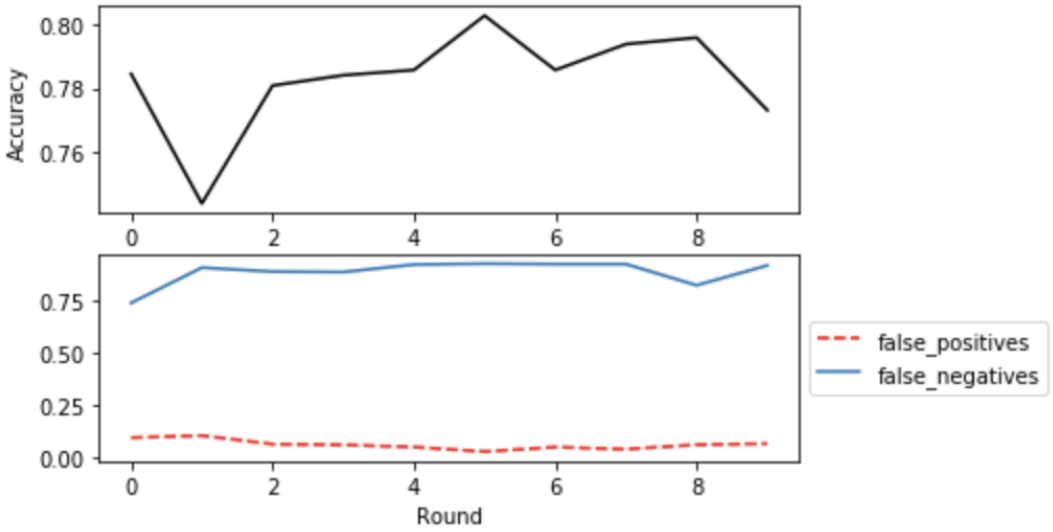

Figure 5.5: Evaluation results of the model

```
Round 1/10
Feature type :  Two Horizontal
Feature (left,top,right,bottom) :  ([[1, 1, 0, 0]], [[1, 1, 0, 0]])
Threshold :  0.0


Performance Evaluation
Accuracy :  0.8091386979377274
False Positive Rate :  0.0
False Negative Rate :  1.0
```

Figure 5.6: Training results of the model (false negative) after Round 1

```
Round 3/10
Feature type :  Two Horizontal
Feature (left,top,right,bottom) :  ([[1, 1, 0, 0]], [[1, 1, 0, 0]])
Threshold :  0.0


Performance Evaluation
Accuracy :  0.8091386979377274
False Positive Rate :  0.0
False Negative Rate :  1.0
```

Figure 5.7: Training results of the model (false negative) after Round 3

When the empirical error is replaced with a formula that would take into consideration the false positive and false negative ratio, we can see that the model is not learning anything new in each of the rounds after the first round because of the weight updation mechanism that is modified. We update the weights using the false positive (false negative) ratio instead of the empirical error. At first we try to see how the model behaves if we try to use only false negatives from the formula shown below in the learning technique. Thus we set $lambda = 1$ in our first experiment.

$$error = lambda * falsenegatives + (1 - lambda) * falsepositives$$

The results are as shown in figures 5.6, 5.7, 5.8, 5.9. The feature chosen by the model is as shown in Figure 5.10. The evaluation results of the model after every iteration is

```
Round 5/10
Feature type :  Two Horizontal
Feature (left,top,right,bottom) :  ([[1, 1, 0, 0]], [[1, 1, 0, 0]])
Threshold :  0.0


Performance Evaluation
Accuracy :  0.8091386979377274
False Positive Rate :  0.0
False Negative Rate :  1.0
```

Figure 5.8: Training results of the model (false negative) after Round 5

```
Round 10/10
Feature type :  Two Horizontal
Feature (left,top,right,bottom) :  ([[1, 1, 0, 0]], [[1, 1, 0, 0]])
Threshold :  0.0


Performance Evaluation
Accuracy :  0.8091386979377274
False Positive Rate :  0.0
False Negative Rate :  1.0
```

Figure 5.9: Training results of the model (false negative) after Round 10

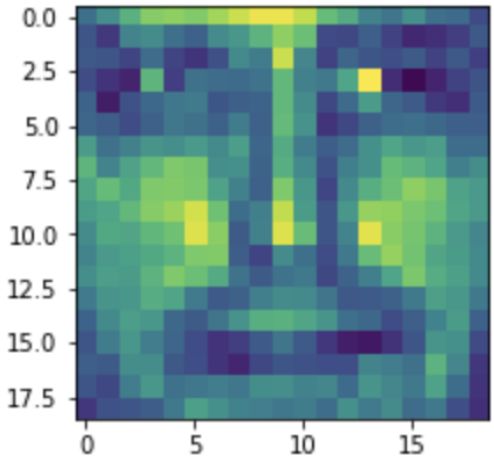

Figure 5.10: Top feature chosen by the model (false negative)

as shown in the Figure 5.11

When we set $lambda = 0$ in the above mentioned formula, the false positive ratio is considered for choosing the weak classifier at every iteration. The results as as shown in the figures 5.12, 5.13, 5.14 and 5.15. The feature chosen by the model is as shown in figures 5.16 and 5.17. The evaluation results of the model after every iteration is as shown in the Figure 5.18.

6 Discussion:

The performance drop of the model with empirical error at the 10th round might be due to overfitting and the skewness in the training data which had 500 images for positive class and 2000 images for the negative class. The weight updation of the model does not change much when we use false positive/false negative as the criteria. Thus we can see a stable performance from the second round when the empirical error criteria is replaced with false positives/false negatives. Though the performance while using false positives

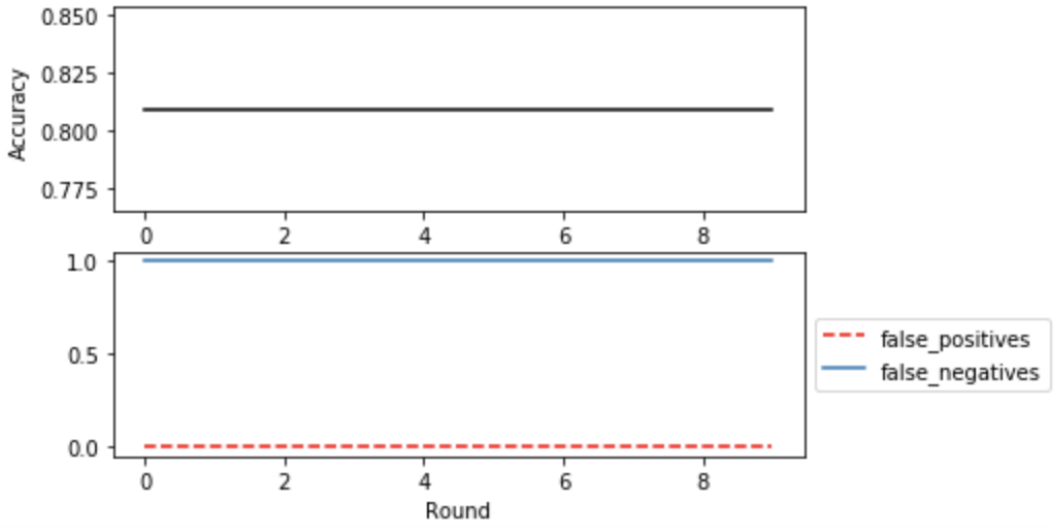

Figure 5.11: Evaluation results of the model (false negative)

```
Round 1/10
Feature type :  Three Horizontal
Feature (left,top,right,bottom) :  ([[11, 11, 2, 1]], [[13, 11, 2, 1], [9, 11, 2, 1]])
Threshold :  -68.0


Performance Evaluation
Accuracy :  0.7808329963606955
False Positive Rate :  0.038980509745127435
False Negative Rate :  0.9830508474576272
```

Figure 5.12: Training results of the model (false positive) after Round 1

```
Round 3/10
Feature type :  Four Rectangle
Feature (left,top,right,bottom) :  ([[2, 1, 1, 2], [1, 3, 1, 2]], [[1, 1, 1, 2], [2, 3, 1, 2]])
Threshold :  196.0


Performance Evaluation
Accuracy :  0.8091386979377274
False Positive Rate :  0.0
False Negative Rate :  1.0
```

Figure 5.13: Training results of the model (false positive) after Round 3

```
Round 5/10
Feature type :  Four Rectangle
Feature (left,top,right,bottom) :  ([[2, 1, 1, 2], [1, 3, 1, 2]], [[1, 1, 1, 2], [2, 3, 1, 2]])
Threshold :  196.0


Performance Evaluation
Accuracy :  0.8091386979377274
False Positive Rate :  0.0
False Negative Rate :  1.0
```

Figure 5.14: Training results of the model (false positive) after Round 5

```
Round 10/10
Feature type :  Four Rectangle
Feature (left,top,right,bottom) :  ([[2, 1, 1, 2], [1, 3, 1, 2]], [[1, 1, 1, 2], [2, 3, 1, 2]])
Threshold :  196.0


Performance Evaluation
Accuracy :  0.8091386979377274
False Positive Rate :  0.0
False Negative Rate :  1.0
```

Figure 5.15: Training results of the model (false positive) after Round 10

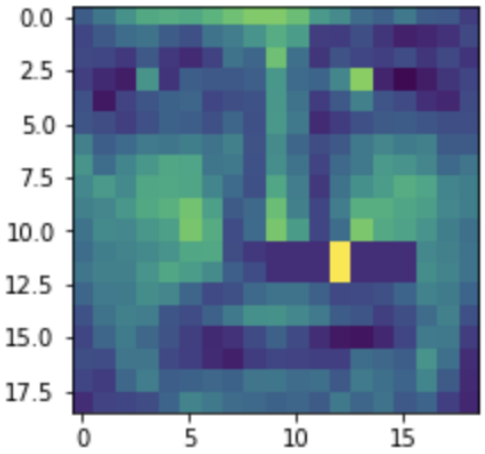

Figure 5.16: Top feature chosen by the model (false positive) after round 1

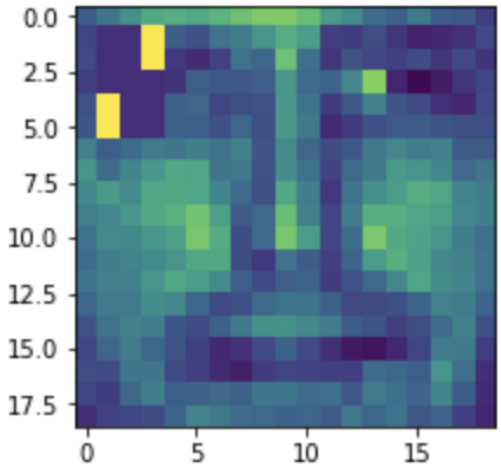

Figure 5.17: Top feature chosen by the model (false positive) after round 10

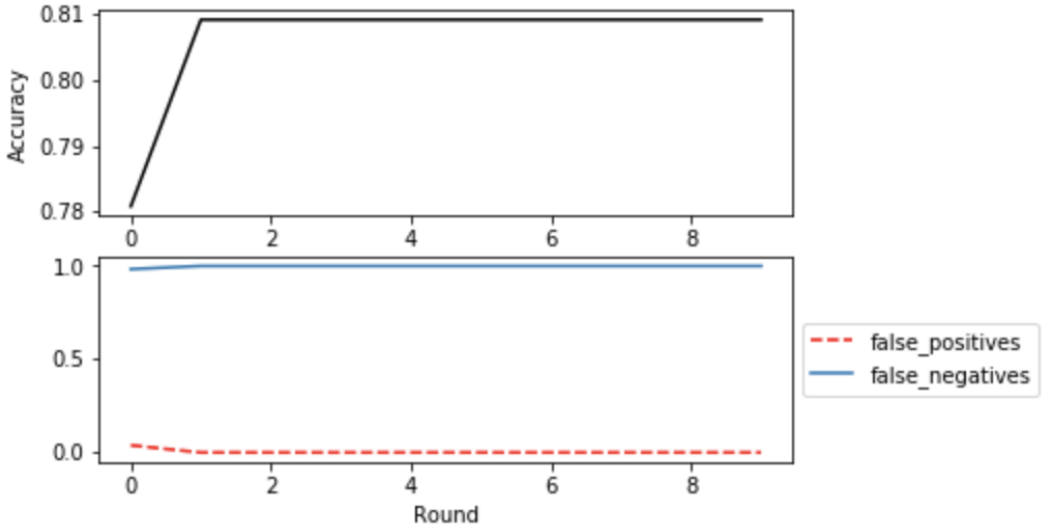

Figure 5.18: Evaluation results of the model (false positive)

was not as expected, we can see from the Figure 5.18 that the feature chosen by the model is an important region of the face (part of the eye). Changing the training and weight updation technique might result in a better performance.

7 Link to project documents:

https://github.com/Shruthi-Sampathkumar/Pattern-Recognition