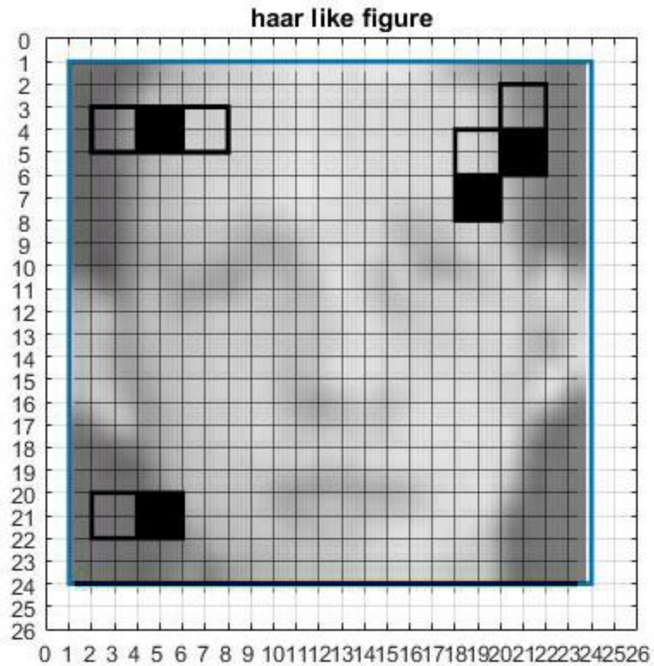
ADVANCED COMPUTER VISION

VIOLA AND JONES
ADABOOST OBJECT DETECTION

Design and implementation of a boosting technique for object identification



Máster en
Automática y Robótica

José Ramón Raindo Portillo

INDEX

INTRODUCTION

VIOLA AND JONES ALGORITHM

- Big development in the last decade
- Used in all kind of devices and media cameras, cellphones, websites
- Viola and Jones AdaBoost object recognition algorithm
  - Fast – (0.7s 10 years ago computers)
  - Low false positive rate
  - Designed for faces, but can be used with all kind of objects

INTRODUCTION

OBJECT RECOGNITION

- Must be trained
- It is composed by 3 stages

TRAINING

Feature extraction

We extract all the possible **Haar-Like** features from a 24x24 window

INTRODUCTION

OBJECT RECOGNITION

- Must be trained
- It is composed by 3 stages

TRAINING

Feature extraction

Weak classifiers creation

Using the Haar-like features, weak classifiers are created

INTRODUCTION

OBJECT RECOGNITION

- Must be trained
- It is composed by 3 stages

TRAINING

- **Feature extraction**
- **Weak classifiers creation**
- **Selection of a set of weak classifiers**

Using a boosting algorithm we select a set of weak classifiers

INTRODUCTION

OBJECT RECOGNITION

- Must be trained
- It is composed by 3 stages

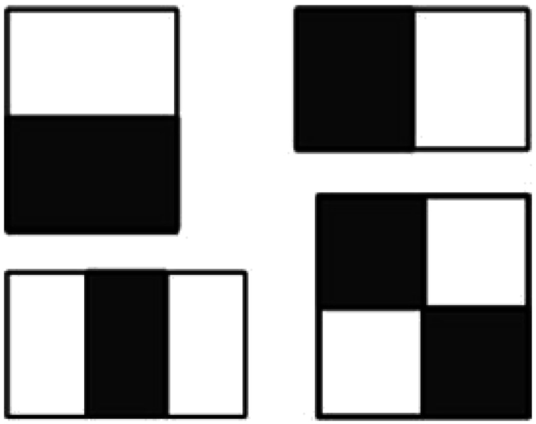
TRAINING

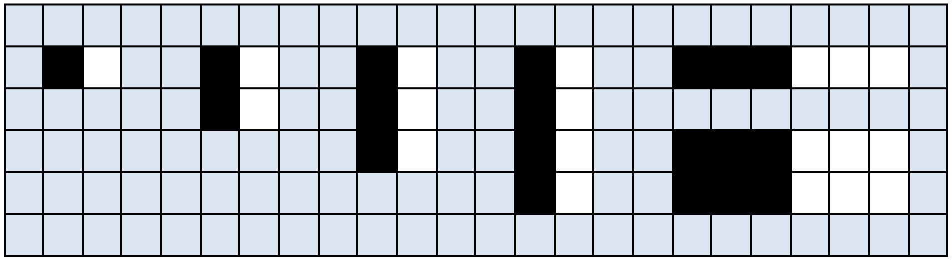
- Feature extraction
- Weak classifiers creation
- Selection of a set of weak classifiers

→ **STRONG CLASSIFIER**

HAAR-LIKE FEATURES

- Set of positive and negative features
- Based in Haar-like figures (black and white squares) inside a 24x24 window:
- A lot of combinations (58140)





HAAR-LIKE FEATURES

10x7 window

2	5	6	4	4	1	7	2	5	6
4	6	4	2	5	8	5	6	4	2
2	5	1	5	1	5	3	5	1	5
2	1	6	4	2	5	8	6	4	2
6	4	5	1	5	1	5	5	1	5
5	1	1	2	5	6	4	1	2	5
1	2	9	4	6	5	5	9	4	6

- The squares represent an operation with the luminance inside an image (the window)

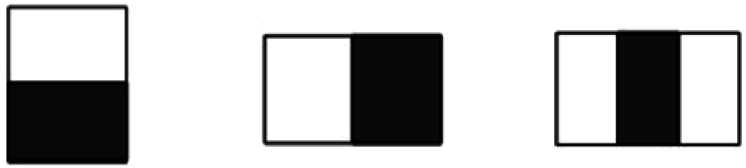
Black part: 2+5+5+1+4+2+1+5+2+5=32

White part: 8+5+5+3+5+8+1+5+6+4=50

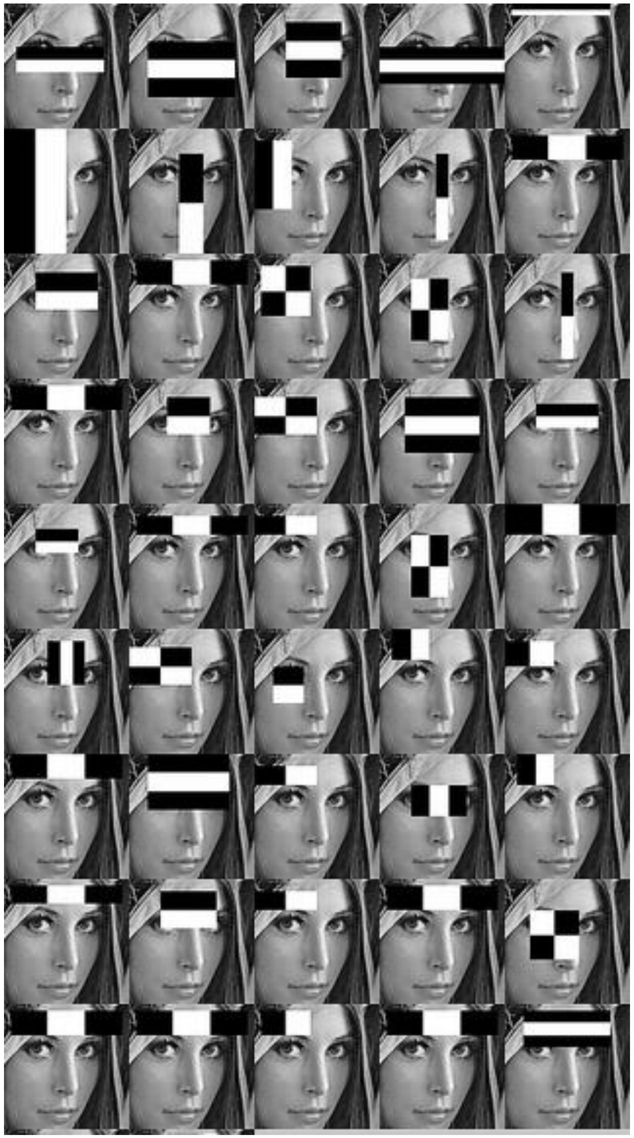
value: 50-32=18

HAAR-LIKE FEATURES

- There are a lot of combinations!
  - each combination is a feature
- Some limitations can be added:
  - Size of the window – 24x24 pixels
  - Limit the size of the Haar-like rectangles (e.g. no smaller than 8x8 pixels)
  - Number of features (In this project we used only 3 types)



- After that, we have 58140 features



HAAR-LIKE FEATURES

- Simple and fast way to compare features
- A lot of features (depend on the window size)
- Haar-like value must be **calculated for every image and feature**
  - Computationally expensive: **O(n)**
  - Important in the boosting part
  - Training over a large pool of images

Solution: INTEGRAL IMAGES

INTEGRAL IMAGES

- New image based on another image
- All pixels are the sum of intensity of the pixels that are above and to the left of it
- It is similar to Haar-like assigned number

Original

5	**2**	**3**	4	1
1	**5**	**4**	2	3
2	2	1	3	4
3	5	6	4	5
4	1	3	2	6

Integral

5	7	10	14	15
6	13	**20**	26	30
8	17	25	34	42
11	25	39	52	65
15	30	47	62	81

$$5 + 2 + 3 + 1 + 5 + 4 = 20$$

INTEGRAL IMAGES

Original

5	2	3	4	1
1	**5**	**4**	**2**	3
2	**2**	**1**	**3**	4
3	5	6	4	5
4	1	3	2	6

5 + 4 + 2 +
2 + 1 + 3 = 17

Integral

5	7	10	**14**	15
6	13	20	26	30
8	17	25	**34**	42
11	25	39	52	65
15	30	47	62	81

34 - 14 - 8 + 5 = 17

- We can calculate a rectangle in an easy way
- Very useful to calculate the Haar-like assigned value!

INTEGRAL IMAGES

Original

5	2	3	4	1
1	**5**	**4**	**2**	3
2	**2**	**1**	**3**	4
3	5	6	4	5
4	1	3	2	6

Integral

5	7	10	**14**	15
6	13	20	26	30
8	17	25	**34**	42
11	25	39	52	65
15	30	47	62	81

As many operations as pixels

$$5+4+2+2+1+3=17$$

$$34-14-8+5=17$$

4 operations ALWAYS

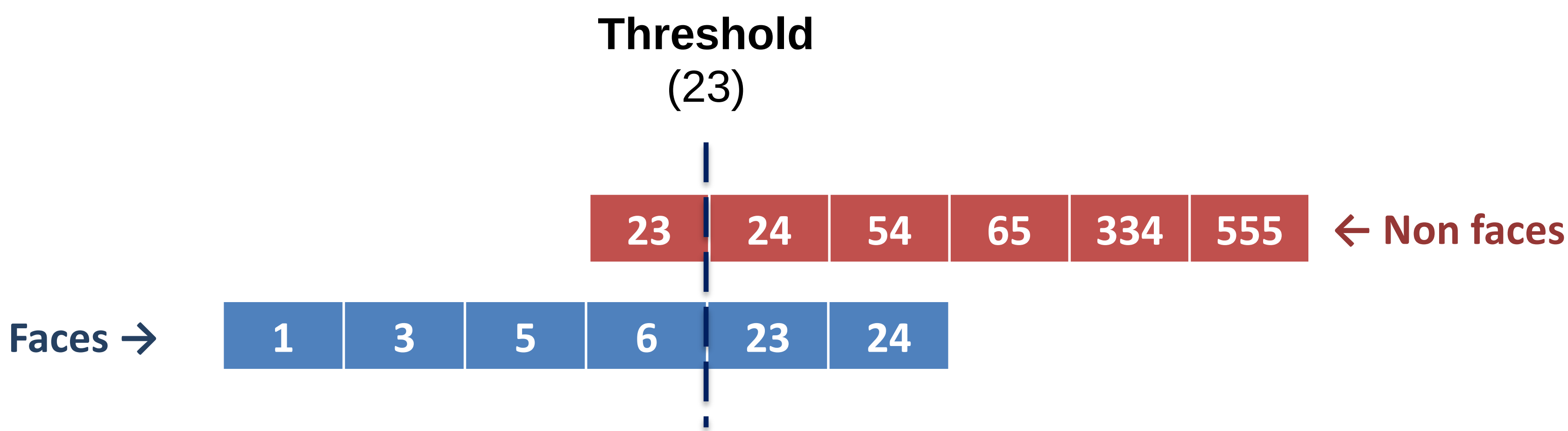
- We can calculate a rectangle in an easy way
- Very useful to calculate the Haar-like assigned value!

INTEGRAL IMAGES

- Viola and Jones use integral images instead of images
- Can be seen as “catching” – the bulk of operations is **done once** and **reused** every time
- Can be saved in an easy way

CLASSIFICATION

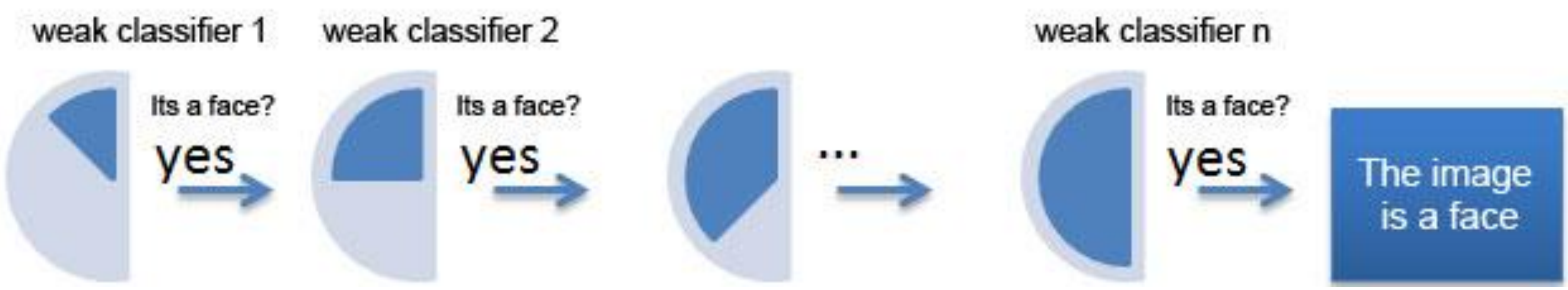
- We make a weak classifier from each Haar Like feature
  - Linear classifier
  - Compares training set Haar like values and creates threshold
  - ‘α’ defines if **faces > threshold** or **faces < threshold**
  - **Marginally better than luck**



Weak classifier: $\begin{cases} (value > 23) \rightarrow face\ (1) \\ (value \leq 23) \rightarrow no\ face\ (-1) \end{cases}$

CLASSIFICATION

- Idea: combine a set of weak classifiers
  - In sequence (cascade): every weak classifier “votes”
  - A good set of weak classifiers will create an strong classifier together



ADABOOST

- Large pool of classifiers: 58.140
- Which ones select
- Order of smallest error of classification?

ADABOOST

- Large pool of classifiers: 58.140
- Which ones select
- Order of smallest error of classification?




ERROR! It will select similar features!

The resulting classifier wont be better than a weak classifier

ADABOOST

- AdaBoost fix this problem by changing the weights
- Similar features will have less weight
- This forces the next feature to be different

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in \mathscr{X}$, $y_i \in \{-1, +1\}$.
Initialize: $D_1(i) = 1/m$ for $i = 1, \ldots, m$.
For $t = 1, \ldots, T$:

- Train weak learner using distribution D_t.
- Get weak hypothesis $h_t : \mathscr{X} \to \{-1, +1\}$.
- Aim: select h_t with low weighted error:

$$\varepsilon_t = \Pr_{i \sim D_t}\left[h_t(x_i) \neq y_i\right].$$

- Choose $\alpha_t = \frac{1}{2} \ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$.
- Update, for $i = 1, \ldots, m$:

$$D_{t+1}(i) = \frac{D_t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution).

Output the final hypothesis:

$$H(x) = \operatorname{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right).$$

Training set: get values of faces and non faces with associated weights

Find this classifier error

Weight of the classifier (depends on the error)

Each image weight is modied: if was succesfully identified, will have less importance next time

The strong classifier

RESULTS

- Training set: 476 images

Positive set: Faces (235 images)

Negative set: No faces (241 images)

RESULTS

- The training took 8h 30’ to finish
- Hard to debug!
- **Some classifiers had negative weight (?)**

- **THE MEANING OF NEGATIVE WEIGHT:**
  - Some of the classifiers are very bad (rate success < luck)
  - Example:

Weak classifier with rate of success of 30% (bad classifier)

Adaboost hypotesis: If we do the OPOSITE - rate of success 70%!!

RESULTS

- Testing set: 510 images

Positive set: Faces (251 images)

Negative set: No faces (259 images)

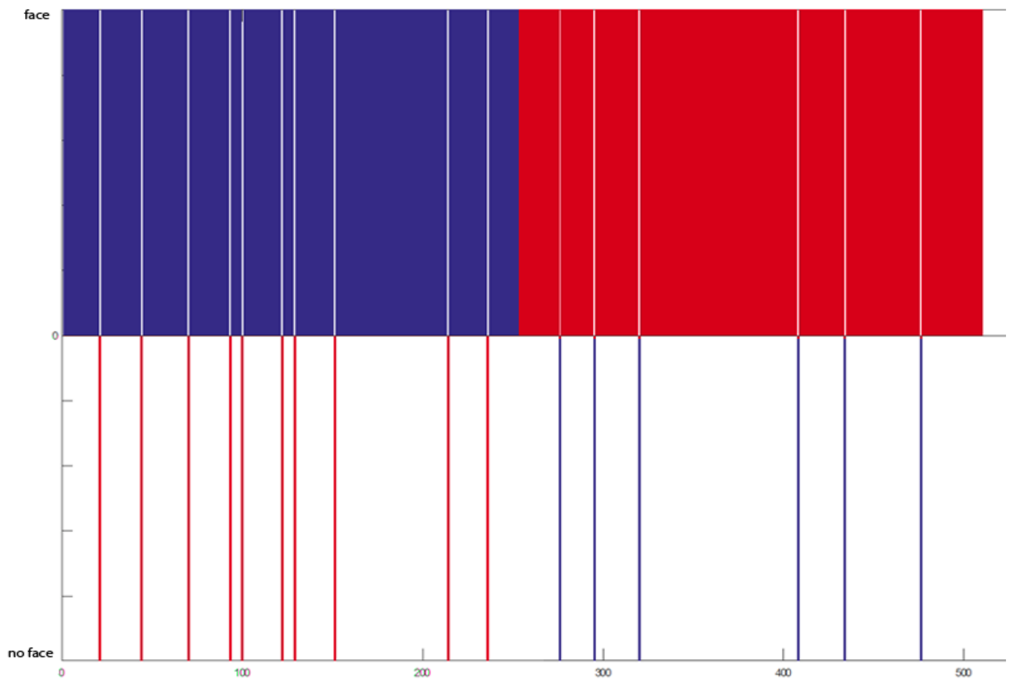
RESULTS

Approach 1:

- Select weak classifiers by the weight absolute value

RESULTS

- Classifier that mix positive and negative weight weak classifiers:

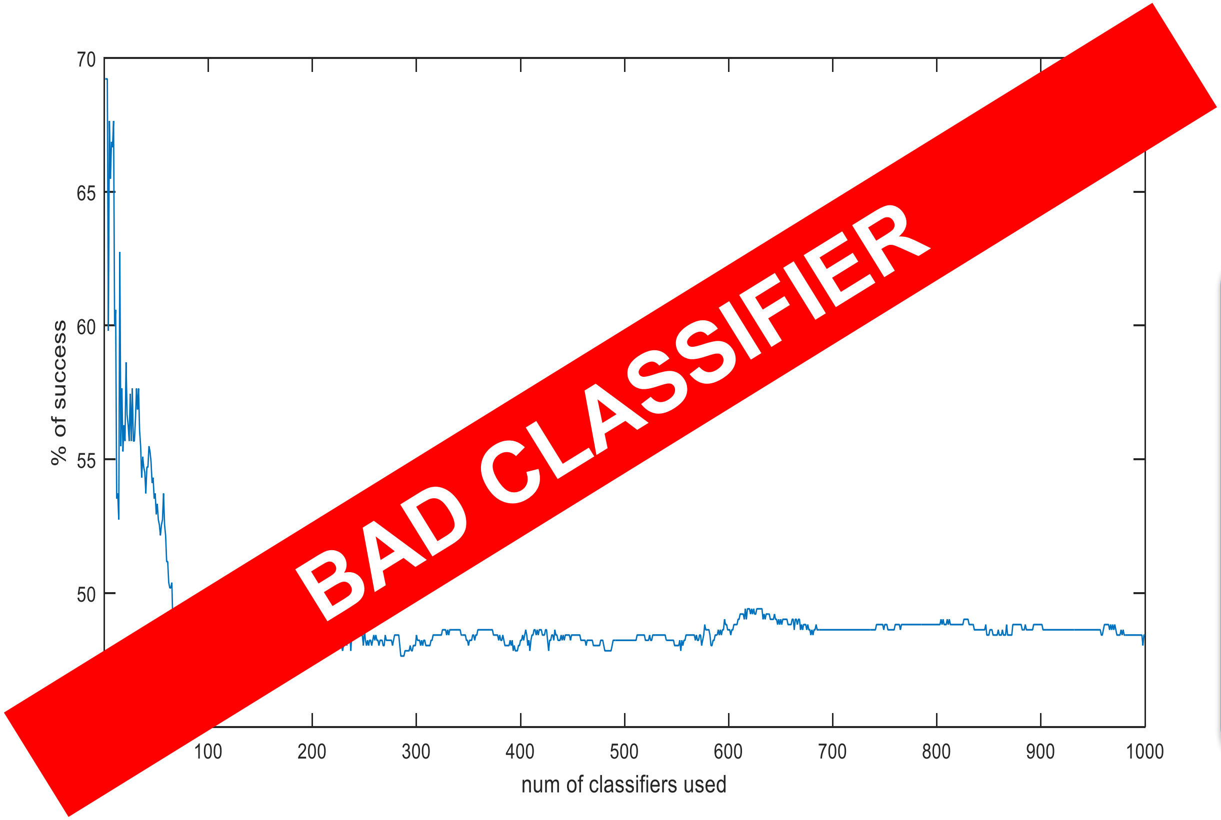


Classification graphic for a 1000 weak classifier strong classifier.

- It almost always take that the value is a face
- Not a good classifier

RESULTS

- Classifier that mix positive and negative weight weak classifiers:



It becomes worse the more classifiers used!!

Possible cause:

The classifiers were created **to minimize errors**

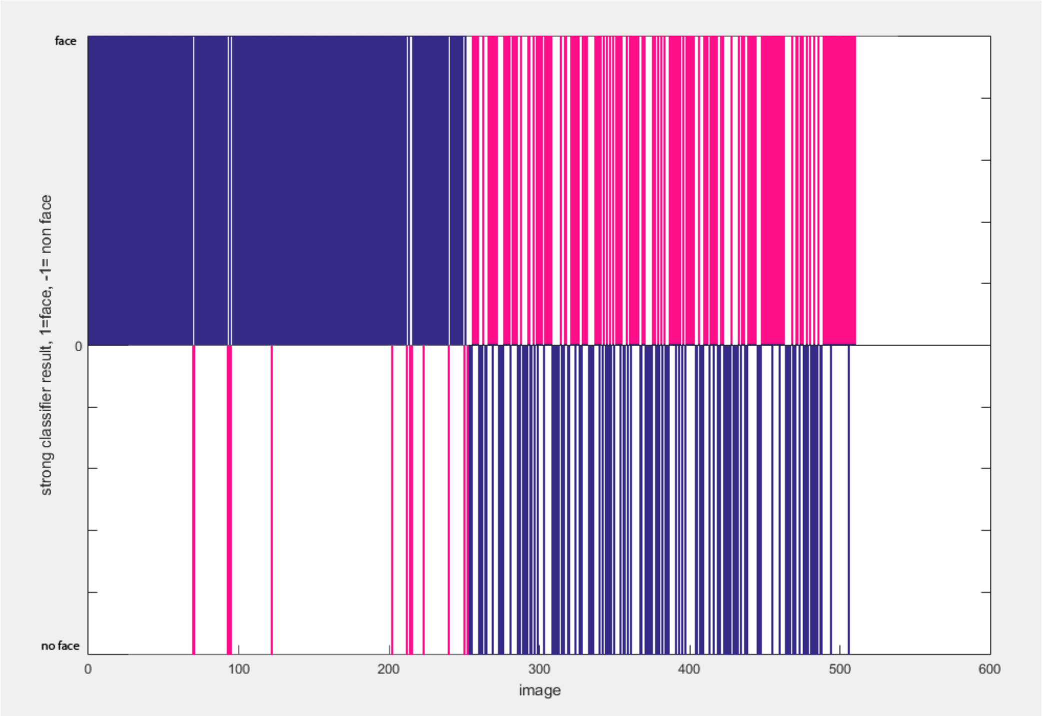
If a classifier has a low rate of success, it means that the **values of this feature are too mixed**

RESULTS

- **Approach 2:**
  - get only the positive weighted classifiers

RESULTS

- Strong classifier with only positive weight weak classifiers



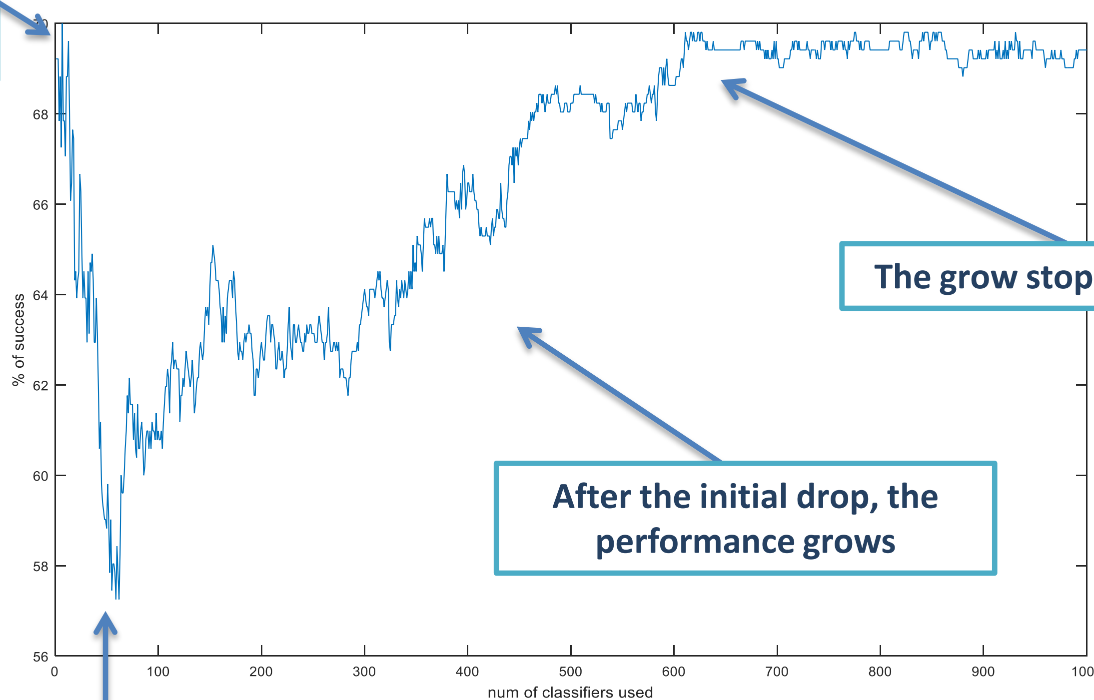
Classification graphic for a 1000 weak classifier strong classifier.

- Is very good identifying faces!
- It finds hard to know when something is not a face

RESULTS

- Strong classifier with only positive weight weak classifiers

The first weak classifiers are strong



The grow stops around 70%

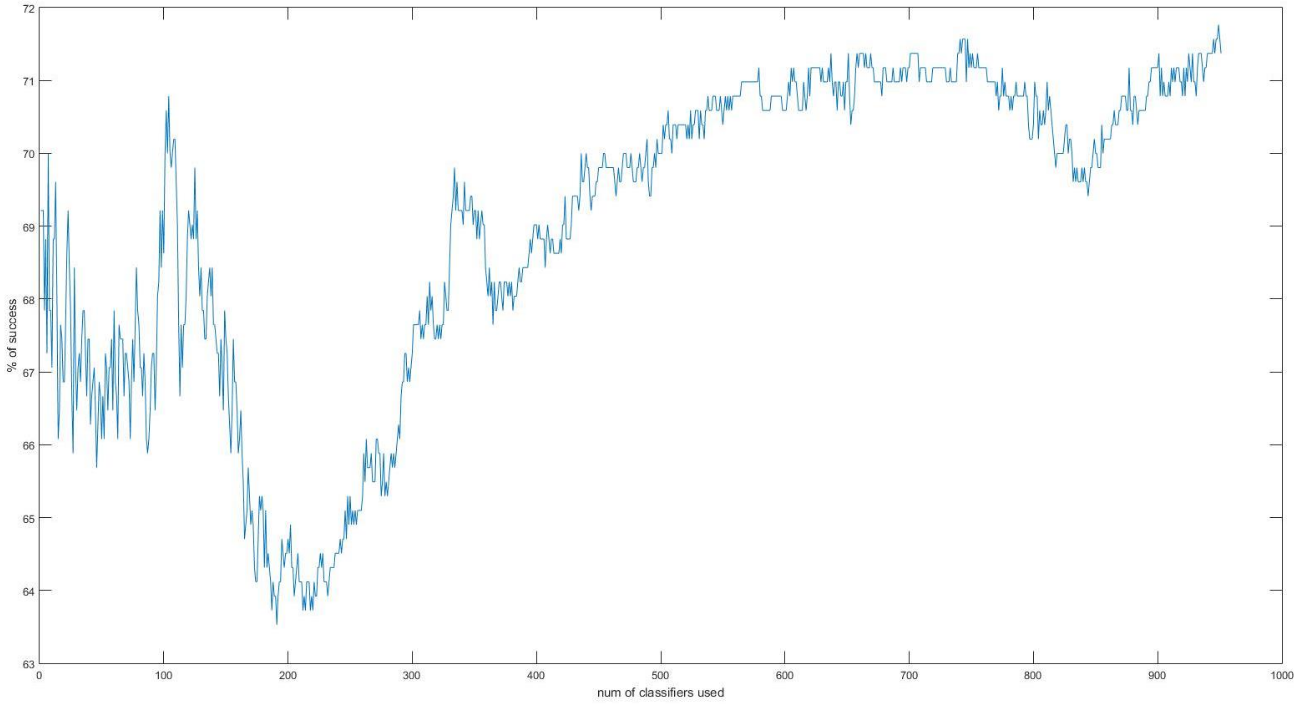
After the initial drop, the performance grows

There are a lot of bad classifiers with high weight!

DOES THE STRONG CLASSIFIER WORKS AS THE BEST OF THE WEAK CLASSIFIERS?

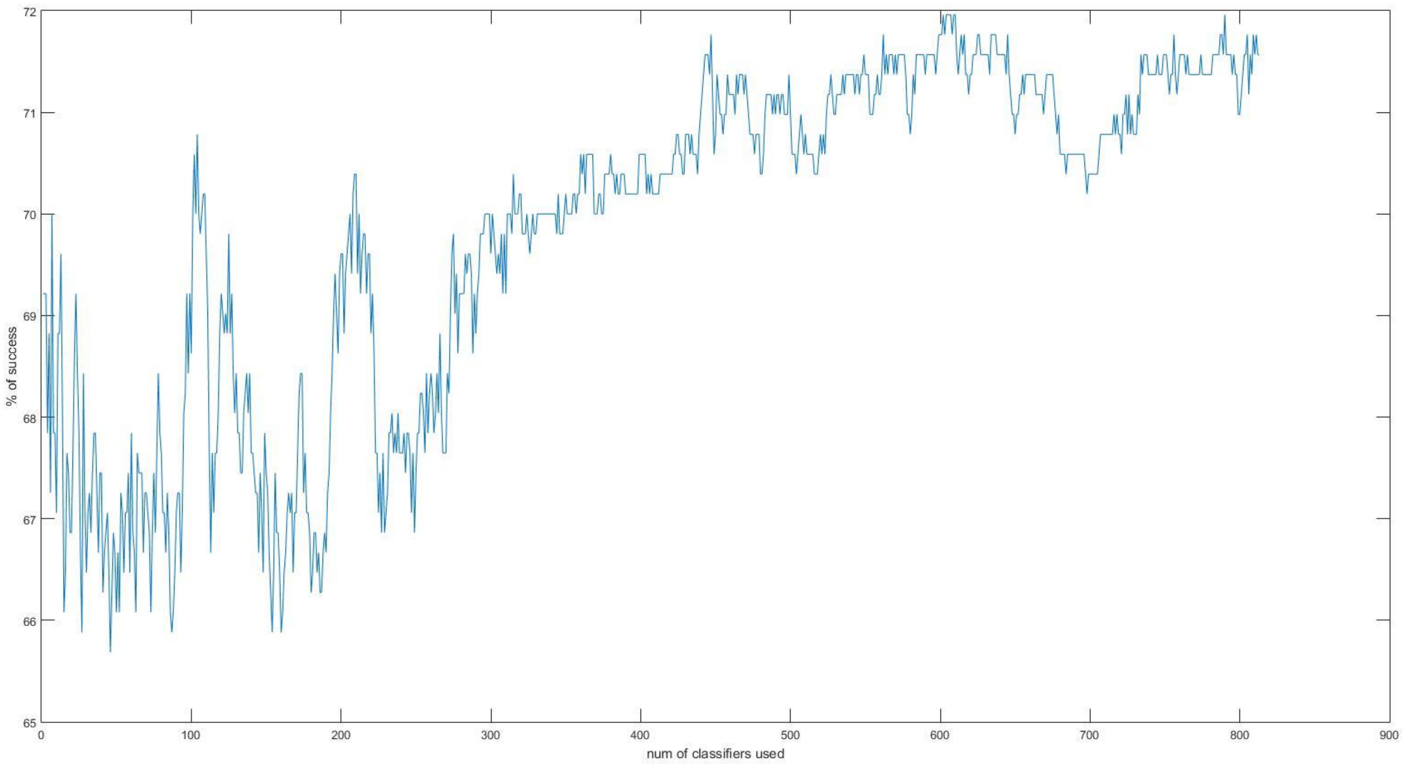
RESULTS

- result after getting ride of the bad classifiers (1) (by hand)



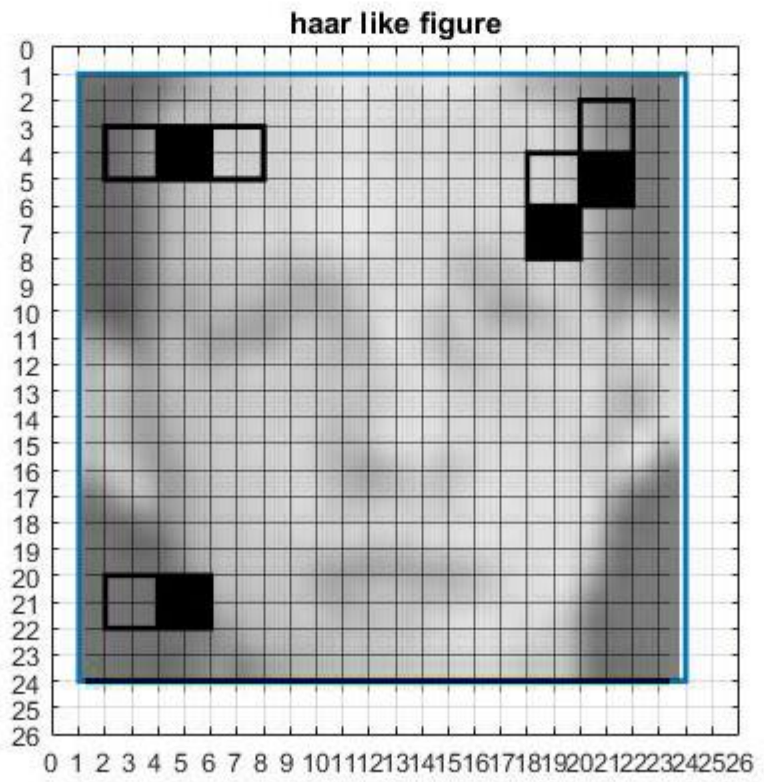
RESULTS

- result after getting rid of the bad classifiers (2) (by hand)



RESULTS

- Example of Haar like feature from some of the best weak classifiers



CONCLUSIONS

- Negative weight weak classifiers aren’t reliable (to much entropic)

- Positive weight weak classifiers work but…
  - Good to identify faces
  - Confused when identifying non faces
  - Not all the high weight classifiers were good

CONCLUSIONS

- How to fix?
  - Improve the best threshold search
  - Increase Haar like features types? (> 8 hours training!)
  - Increase non faces training set? (> 8 hours training!)
  - Use bigger training set? (> 8 hours training!)
  - Repeat the learning process iteratively? (> 8 hours training!)

- Adaboost gives priority to the first weak classifiers it encounters from the same family! (pattern recognition problem)
  - Use a different boosting algorithm?

THANK YOU VERY MUCH!