

Boosting algorithm

Diane Lingrand and Frédéric Precioso



Membre de l'UNIVERSITÉ CÔTE D'AZUR

SI 4

2019 - 2020

1 Boosting algorithm

- Idea
- A first simple example to understand boosting idea
- Probabilistic boosting and Adaboost
- A simple example for understanding Adaboost

2 Application to face detection

1 Boosting algorithm

- Idea
- A first simple example to understand boosting idea
- Probabilistic boosting and Adaboost
- A simple example for understanding Adaboost

2 Application to face detection

1 Boosting algorithm

■ Idea

- A first simple example to understand boosting idea
- Probabilistic boosting and Adaboost
- A simple example for understanding Adaboost

2 Application to face detection

- Supervised learning
 - learning dataset
 - test dataset
- Binary classification

- Separation between data corresponding to some criterions and data not corresponding.



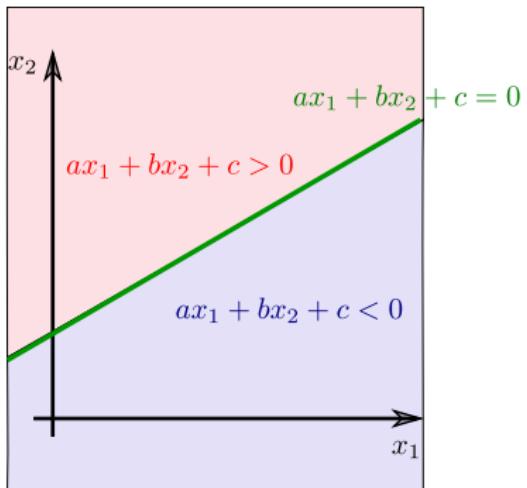
Main idea of boosting

- Build a strong learner from weak learners
- Example :
 - Green corresponds to fir tree.
 - Vertical shapes corresponds to fir tree.
 - Shapes with bottom larger than top corresponds to fir tree.
 - When dominant color is red, it is not a fir tree.
 -
- Example of sport bet and heuristics.
 - We ask experts simple rules that are true more than 50%
 - We focus on examples for which a rule fails and we ask other rules to experts.
 - And loop the process

- Mathematician Kearns asks : “Is it possible to make as good as possible a weak learner” (that is to say better than random) ?
- Schapire, answered in 90 : “ Yes ! ” , and exhibited the first elementary boosting algorithm which shows that a weak binary classifier can always improve by being trained on 3 subsamples well chosen.
- The choice of the weak learner does not have any importance (a decision tree, a bayesian classifier, a SVM, a Neural Network, etc.), but one has to choose the 3 training subsamples with respect to its performance.

An example for a weak classifier : line in 2d plane

- points lie in a 2d plane
- the weak classifier is defined by a line of equation $ax_1 + bx_2 + c = 0$
 - divides the plane into 2 area :
 - $(x_1; x_2)$ for which $ax_1 + bx_2 + c > 0$
 - $(x_1; x_2)$ for which $ax_1 + bx_2 + c < 0$
- learning means to find coefficients a , b , and c .



Elementary boosting algorithm

S : learning data set of m elements

- ① Learn the classifier h_1 on subset $S_1 \subset S$. Test of h_1 on $S \setminus S_1$.
- ② Learn the classifier h_2 on subset $S_2 \subset S \setminus S_1$ with half of elements of S_2 wrongly classified by h_1 .
- ③ Learn classifier h_3 on subset $S_3 \subset S \setminus S_1 \setminus S_2$: contains elements for which rules h_1 and h_2 answer differently.
- ④ H : Majority vote between answers of h_1 , h_2 and h_3 .

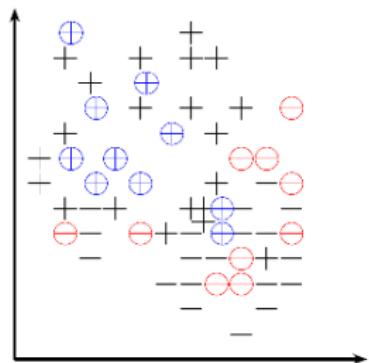
1 Boosting algorithm

- Idea
- A first simple example to understand boosting idea
- Probabilistic boosting and Adaboost
- A simple example for understanding Adaboost

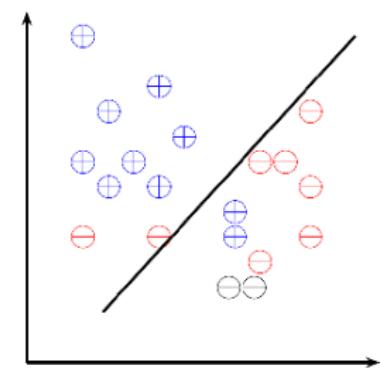
2 Application to face detection

A toy example (1)

Classifiers are lines. Learning a classifier corresponds to find the linear separation of data.

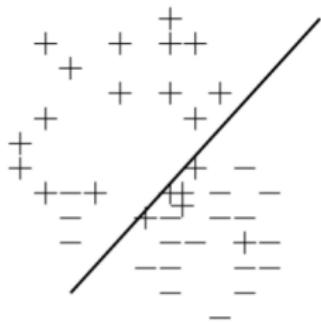


S : set of + and -
 S_1 : reds and blues

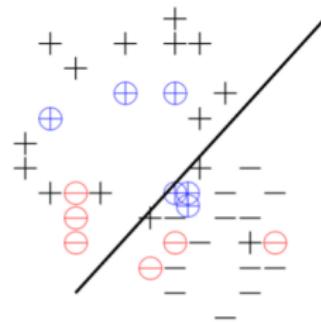


Classifier h_1 learned on S_1 .

A toy example (2)

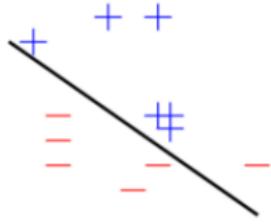


$S \setminus S_1$ et h_1

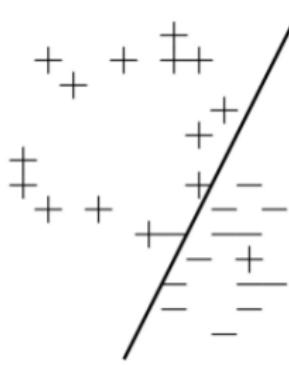


$S_2 \subset S \setminus S_1$: reds and blues

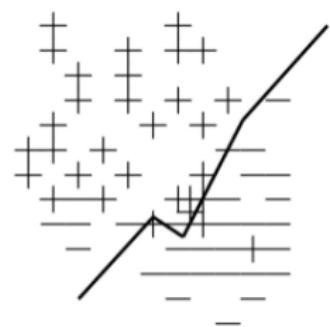
A toy example (3)



h_2 learned on S_2



S_3 and h_3



S and H

1 Boosting algorithm

- Idea
- A first simple example to understand boosting idea
- **Probabilistic boosting and Adaboost**
- A simple example for understanding Adaboost

2 Application to face detection

3 main ideas :

- ① A set of specialized experts and ask them to vote to take a decision.
- ② Adaptive weighting of votes by multiplicative update.
- ③ Modifying example distribution to train each expert, increasing the weights iteratively of examples misclassified at previous iteration.

- S learning dataset
- Initialisation : all samples have same weights ($D_1(i), i \in \{1 \dots m\}$).
- Iterations : for $t \in \{1 \dots T\}$
 - learn h_t by minimisation of an **error**
 - compute **weight** α_t and **weights** $D_t(i)$ (or distribution)
- Compute strong classifier :

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

Adaboost : Adaptable boosting

- $S = \{(x_1, y_1), \dots, (x_m, y_m)\}$ with $x_i \in X$ et $y_i \in \{-1, +1\}$
- Initialisation : $D_1(i) = \frac{1}{m}$ with $i \in \{1 \dots m\}$
- For $t \in \{1 \dots T\}$:
 - find $h_t : X \rightarrow \{-1, +1\}$ minimizing error ϵ_t defined by :

$$\epsilon_t = \sum_{i=1}^m D_t(i)[y_i \neq h_t(x_i)]$$

- compute **weights** :

$$\alpha_t = \frac{1}{2} \ln\left(\frac{1 - \epsilon_t}{\epsilon_t}\right)$$

- compute **distribution** :

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

where Z_t is for normalisation : $1 = \sum_{i=1}^m D_t(i)$

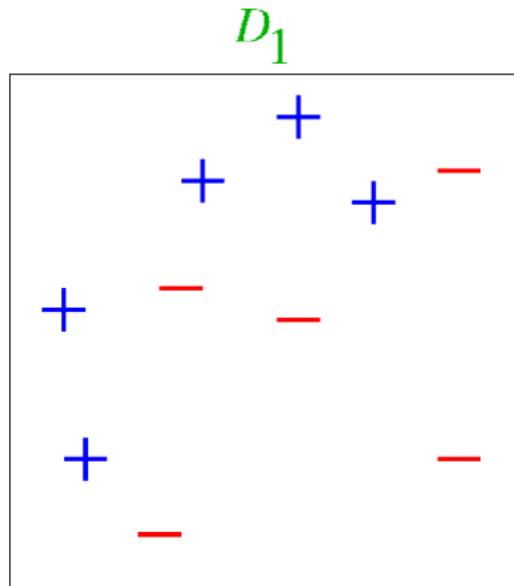
- Compute **strong classifier** :

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

1 Boosting algorithm

- Idea
- A first simple example to understand boosting idea
- Probabilistic boosting and Adaboost
- A simple example for understanding Adaboost

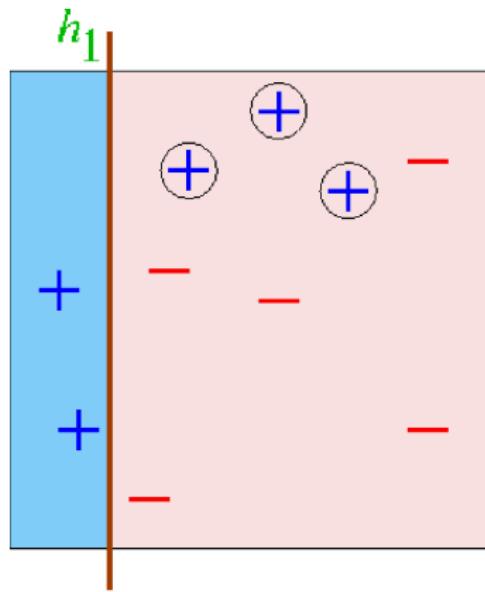
2 Application to face detection



$D_1 :$

0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

Step 1



$D_2 :$

0.167	0.167	0.167	0.071	0.071	0.071	0.071	0.071	0.071	0.071
-------	-------	-------	-------	-------	-------	-------	-------	-------	-------

$$\epsilon_1 = 0.30$$

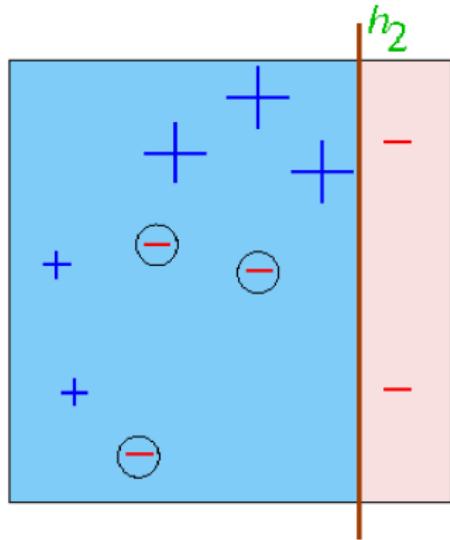
$$\alpha_1 = \frac{1}{2} \ln\left(\frac{1 - \epsilon_1}{\epsilon_1}\right) = \frac{1}{2} \ln\left(\frac{0.7}{0.3}\right) = 0.42$$

$$D_2(1) = \frac{1}{Z_1} D_1(1) e^{\alpha_1} = \frac{0.152}{Z_1}$$

$$D_2(4) = \frac{1}{Z_1} D_1(4) e^{-\alpha_1} = \frac{0.065}{Z_1}$$

$$Z_1 = 3 \times 0.152 + 7 \times 0.065 = 0.911$$

Step 2



$$\epsilon_2 = 0.21$$

$$\alpha_2 = \frac{1}{2} \ln\left(\frac{1 - \epsilon_2}{\epsilon_2}\right) = 0.65$$

$$D_3(1) = \frac{D_2(1)}{Z_2} e^{-\alpha_2} = 0.167 e^{-0.65} = \frac{0.0876}{Z_2}$$

$$D_3(4) = \frac{D_2(4)}{Z_2} e^{-\alpha_2} = 0.071 e^{-0.65} = \frac{0.036}{Z_2}$$

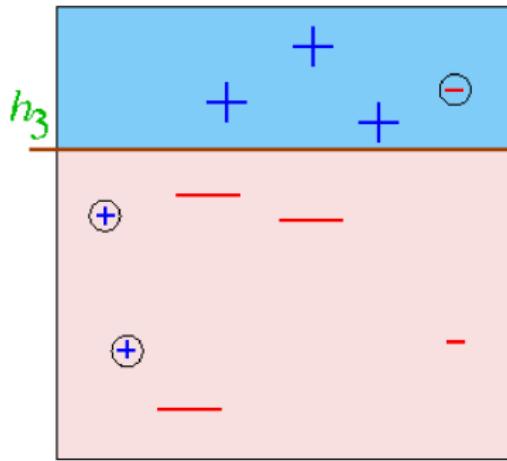
$$D_3(8) = \frac{D_2(8)}{Z_2} e^{+\alpha_2} = 0.071 e^{+0.65} = \frac{0.1357}{Z_2}$$

$$Z_2 = 3 \times 0.0876 + 4 \times 0.036 + 3 \times 0.1357 = 0.814$$

$D_3 :$

0.11	0.11	0.11	0.044	0.044	0.044	0.044	0.17	0.17	0.17
------	------	------	-------	-------	-------	-------	------	------	------

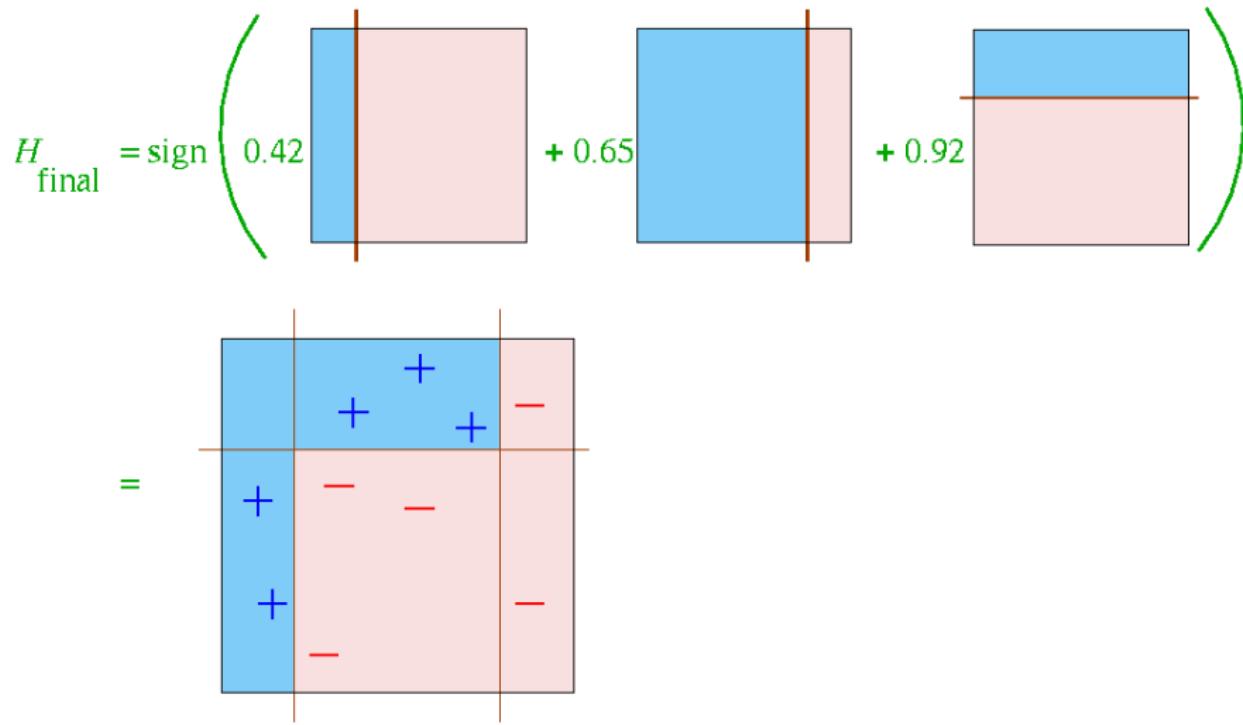
Step 3



$$\epsilon_3 = 3 \times 0.044 = 0.13$$

$$\alpha_3 = 0.92$$

Final classifier



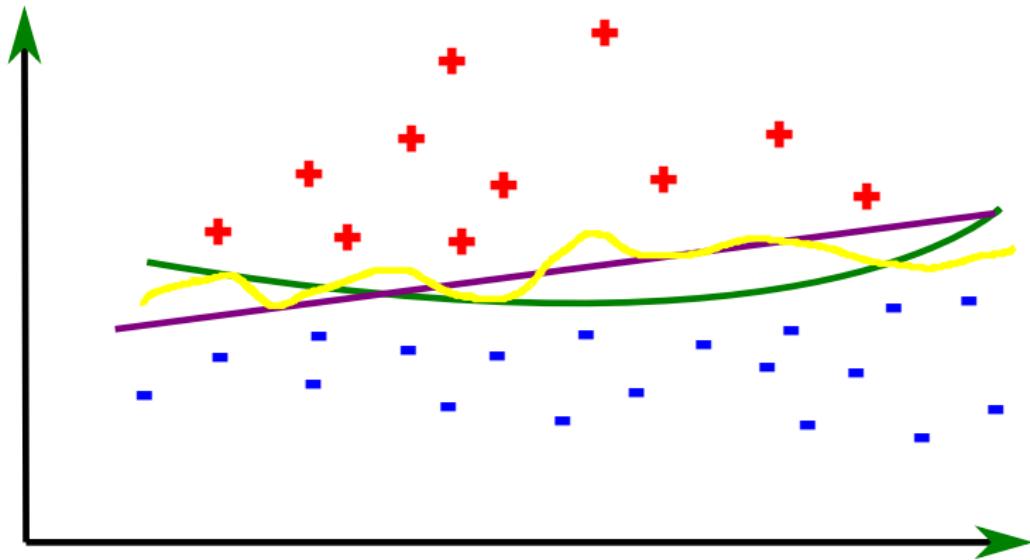
- Advantages :

- Fast
- Easy to implement
- only 1 parameter : number of boosting iteration
- extensible to multi classes classification
- able to detect outliers

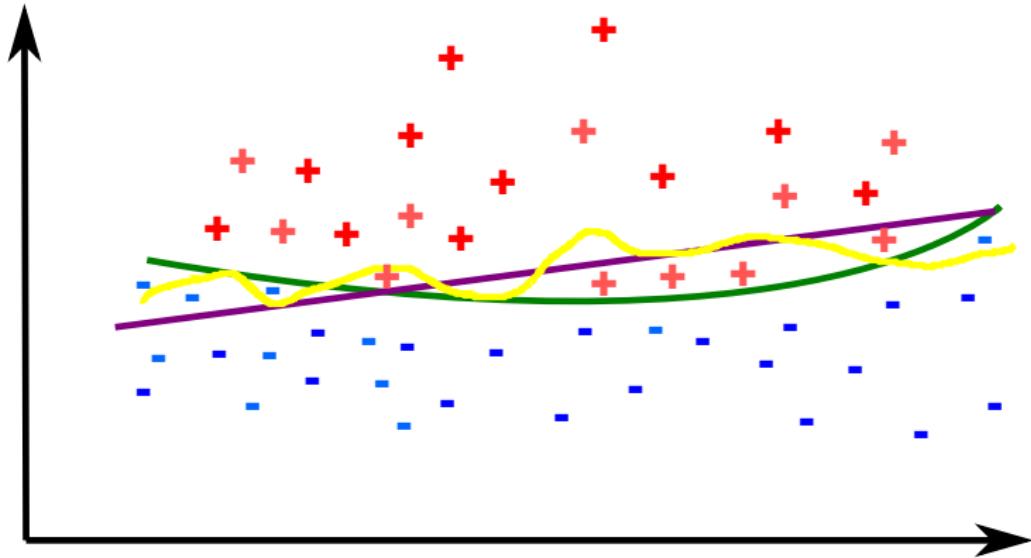
- Drawbacks :

- depends on learning data and weak classifiers
- could fail if :
 - the weak classifier is too complex
 - low margins → overfitting
 - the weak classifier is too weak
 - underfitting
- noise sensitive

Overfitting and Underfitting



Overfitting and Underfitting



- Solve overfitting when noisy data :
 - Weight Decay (1998) : introduces slack variables regularizing the error
 - GentleBoost (1998)
 - LogitBoost (2000) : additive logistic regression model (statistical point of view)
 - Regularized AdaBoost (2000) : “soft” margins
 - BrownBoost (2001) : removes examples many times misclassified, considered as “noisy”
 - WeightBoost (2003) : weights in final decision dependent on input
- Reduce the number of features (or Weak Learner) :
 - FloatBoost (2003) : removes the worst WL after each iteration
 - JointBoost (2004) : multi-class, jointly train N binary classifier sharing same features
- Adaboost Multi-class
 - AdaBoost.M1 : same as standard but $y \in \{1 \dots k\}$
 - AdaBoost.M2 : introduces a confidence degree
 - AdaBoost SAMME (2006)

Multi-class boosting : AdaBoost SAMME

- $S = \{(x_1, y_1), \dots, (x_m, y_m)\}$ with $x_i \in X$ et $y_i \in \{1 \dots k\}$
- Initialisation : $D_1(i) = \frac{1}{m}$ with $i \in \{1 \dots m\}$
- For $t \in \{1 \dots T\}$:
 - find $h_t : X \rightarrow \{1 \dots k\}$ minimizing error ϵ_t defined by :

$$\epsilon_t = \sum_{i=1}^m D_t(i) g(y_i, h_t(x_i)) \text{ with } g(a, b) = 1 \text{ if } a = b \text{ else } 0$$

- compute **weights** :

$$\alpha_t = \frac{1}{2} \ln\left(\frac{1 - \epsilon_t}{\epsilon_t}\right) + \frac{1}{2} \ln(k - 1)$$

- compute **distribution** :

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t g(h_t(i), y_i))}{Z_t} \text{ where } 1 = \sum_{i=1}^m D_t(i)$$

- Compute **strong classifier** :

$$\arg \max_k \left(\sum_{t=1}^T \alpha_t g(h_t(x), k) \right)$$

- Detection, tracking and face recognition
- Spam, Zip Code OCR
- Text classification : Schapire and Singer
- OCR : Schwenk and Bengio (neural networks)
- Natural language Processing : Collins ; Haruno, Shirai and Ooyama
- Image retrieval : Thieu and Viola
- Medical diagnosis : Merle et al.
- Fraud Detection : Rätsch & Müller 2001
- Drug Discovery : Rätsch, Demiriz, Bennett 2002
- Elect. Power Monitoring : Onoda, Rätsch & Müller 2000

Example using scikit-learn

From <https://scikit-learn.org/stable/modules/ensemble.html#gradient-tree-boosting> :

```
from sklearn.model_selection import cross_val_score
from sklearn.datasets import load_iris
from sklearn.ensemble import AdaBoostClassifier

iris = load_iris()
clf = AdaBoostClassifier(n_estimators=100)
scores = cross_val_score(clf, iris.data, iris.target, cv=5)
scores.mean()
```

1 Boosting algorithm

- Idea
- A first simple example to understand boosting idea
- Probabilistic boosting and Adaboost
- A simple example for understanding Adaboost

2 Application to face detection

Face detection







The image is a composite of two parts. On the left, a woman with glasses and dark hair is speaking into a black microphone. On the right, there is a presentation slide with a dark purple header containing the text "A deeper view of Deep Learning". Below the header, the names "Mélanie Ducoffe, Frédéric Precioso" are listed. Underneath that, the text "Laboratoire i3S - UNS CNRS UMR 7271" is displayed. At the bottom of the slide, the date "06/07/2016" is shown. The footer of the slide features several logos: "CNRS" in a red circle, "i3S" in blue with yellow lines, and "Université Paul-Sabatier Toulouse" in a green and white logo.

A red square box is drawn around the top-left corner of the image area.



M

A deeper view of Deep Learning

Mélanie Ducoffe, Frédéric Precioso

Laboratoire I3S - UNS CNRS UMR 7271

06/07/2016

E

CNRS

i3S

Université
Nice
Savoie
Toulon



The image is a composite of two parts. On the left, a woman with glasses and brown hair is speaking into a black microphone. A red square box is drawn around her head area. On the right, there is a presentation slide with a dark blue header containing the white text "A deeper view of Deep Learning". Below the header, the names "Mélanie Ducoffe, Frédéric Precioso" are listed. Further down, the text "Laboratoire I3S - UNS CNRS UMR 7271" is displayed, followed by the date "06/07/2016". At the bottom of the slide, there are logos for "CNRS", "I3S", and "Université Nice Sophia Antipolis".

M

E

A deeper view of Deep Learning

Mélanie Ducoffe, Frédéric Precioso

Laboratoire I3S - UNS CNRS UMR 7271

06/07/2016

CNRS

i3S

Université
Nice Sophia Antipolis



The image is a composite of two parts. On the left, a woman with glasses and dark hair is speaking into a black microphone. A red rectangular box highlights the top-left corner of the image. On the right, there is a presentation slide with a dark purple header containing the text "A deeper view of Deep Learning". Below the header, the names "Mélanie Ducoffe, Frédéric Precioso" are listed. Further down, the text "Laboratoire i3S - UNS CNRS UMR 7271" is displayed, followed by the date "06/07/2016". At the bottom of the slide, there are logos for CNRS, i3S, and Université Paul-Sabatier Toulouse.

A deeper view of Deep Learning

Mélanie Ducoffe, Frédéric Precioso

Laboratoire i3S - UNS CNRS UMR 7271

06/07/2016

CNRS i3S Université Paul-Sabatier Toulouse



The image is a composite of two parts. On the left, a woman with glasses and brown hair is speaking into a black microphone. A red rectangular box highlights a specific area on her forehead. On the right, there is a presentation slide with a dark blue header containing the white text "A deeper view of Deep Learning". Below the header, the names "Mélanie Ducoffe, Frédéric Precioso" are listed. Further down, the text "Laboratoire I3S - UNS CNRS UMR 7271" is displayed, followed by the date "06/07/2016". At the bottom of the slide, there are logos for CNRS (a purple circle), I3S (a purple stylized 'i3S' with yellow lines), and Université Grenoble Alpes (a logo with a green 'U' and blue text).

A deeper view of Deep Learning

Mélanie Ducoffe, Frédéric Precioso

Laboratoire I3S - UNS CNRS UMR 7271

06/07/2016

CNRS

i3S

Université
Grenoble
Alpes

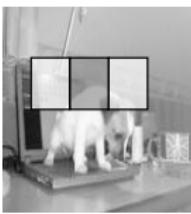
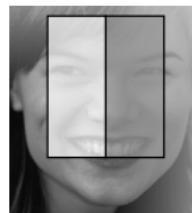
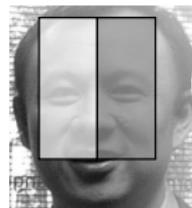
- slide a window across image and evaluate a face model at every location.
- sliding window detector must evaluate tens of thousands of location/scale combinations.
- faces are really rare compare to the number of evaluations
 - for computational efficiency, we should try to spend as little time as possible on the non-face windows
 - false positive rate should be very low ($< 10^{-6}$ for an image of 1 Mpixels)

- real-time object detection
- slow training but fast detection
- 3 main ideas :
 - integral images for fast feature evaluation
 - boosting for feature selection
 - cascade classifiers for fast rejection of non face

[CVPR 2001] : P. Viola and M. Jones. "Rapid object detection using a boosted cascade of simple features".

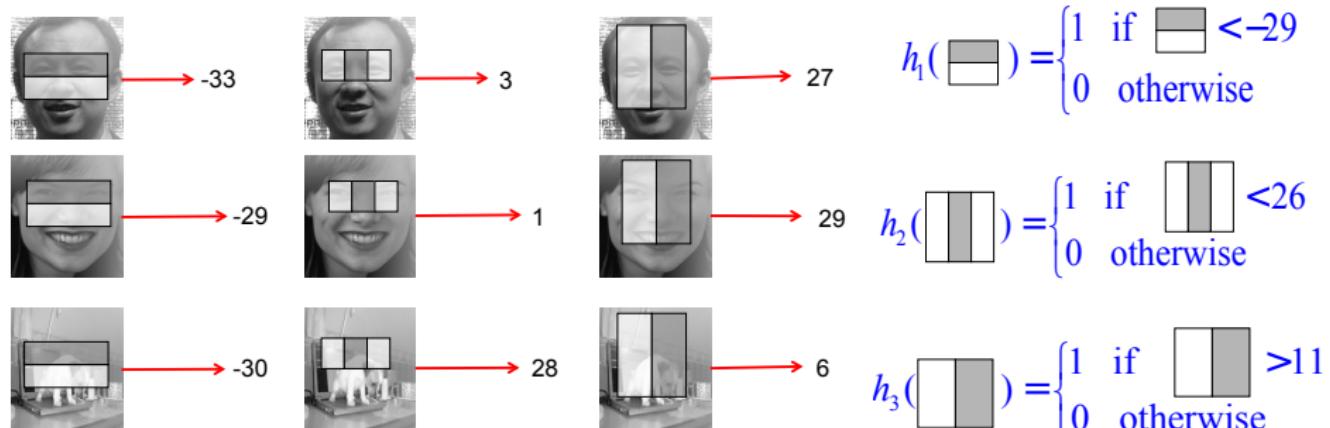
[IJCV 2004] : P. Viola and M. Jones. "Robust real-time face detection" , 57(2).

Finding weak classifiers for faces



$$f_t(W) = \sum_{\text{blanc}} \text{pixels} - \sum_{\text{noir}} \text{pixels}$$

How to learn classifiers ?



Generalisation :

$$h_t(W) = \begin{cases} 1 & \text{si } p_t f_t(W) > p_t \theta_t \\ 0 & \text{sinon} \end{cases}$$

where the parity p_t equals ± 1 , threshold θ_t has to be learned.

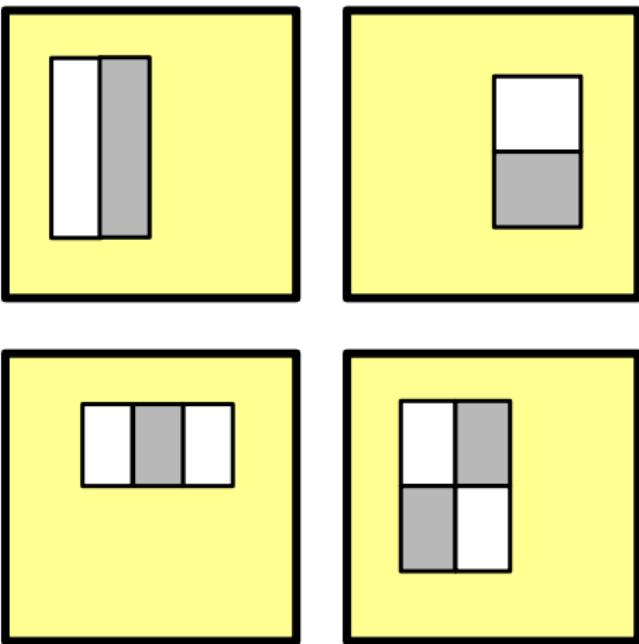
- S learning dataset : faces and non faces
- Initialisation : same weights for faces ($\frac{1}{m_v}$) and same weights for non faces ($\frac{1}{m_{nv}}$).
- Iterations : for $t \in \{1 \dots T\}$
 - for all available filters :
 - learn threshold θ for all learning samples
 - select best filter h_t with threshold θ_t
 - compute weights α_t and $D_t(i)$
 - compute strong classifier

K : number of filters

Learning complexity : $O(T.(nb. samples).K)$

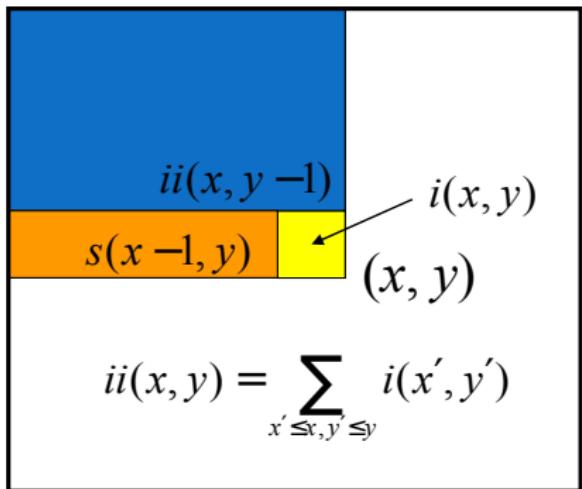
Weak classifiers : rectangular filters

- based on Haar filters
- difference between sum of white related pixels and sum of black related pixels
- 3 types : 2, 3 or 4 rectangles
- combinaisons (rectangles, sizes, positions) for a 24x24 window : 160 000 filters



Integral Image (II) computation

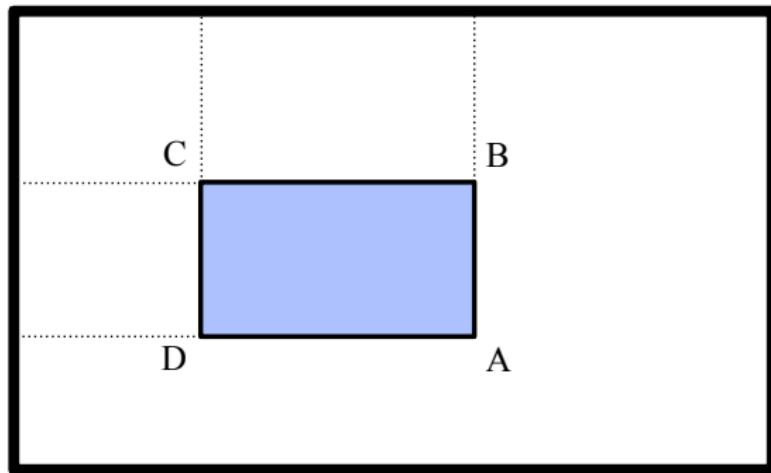
- i is the initial image; ii is the integral image
- Each pixel (x,y) of the integral image represents the sum of all pixels above and to the left of (x,y) , from the initial image.
- Integral image is computed in one pass using a temporary structure representing the sum of pixels on the same row, at left : s .



$$\begin{aligned}s(x, y) &= s(x - 1, y) + i(x, y) \\ ii(x, y) &= ii(x, y - 1) + s(x, y)\end{aligned}$$

Use of integral image

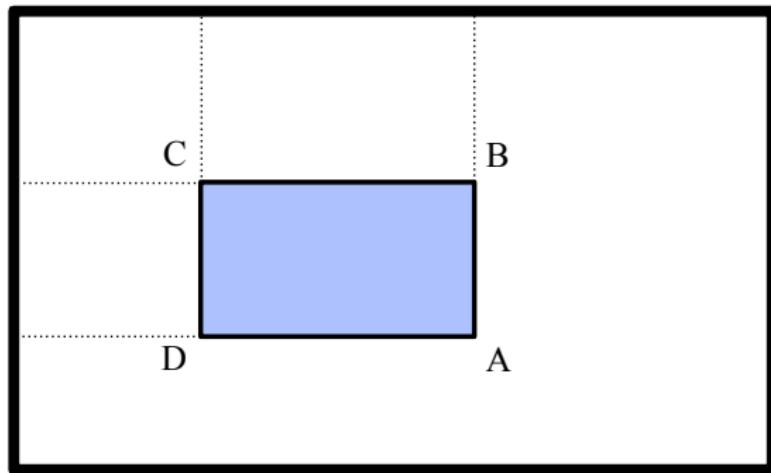
Compute the sum of pixels inside a rectangular area ABCD :



$$S = \sum_{x=x_C}^{x_A} \sum_{y=y_D}^{y_C} i(x, y) = ii_A - ii_B + ii_C - ii_D$$

Use of integral image

Compute the sum of pixels inside a rectangular area ABCD :



$$S = \sum_{x=x_C}^{x_A} \sum_{y=y_D}^{y_C} i(x, y) = ii_A - ii_B + ii_C - ii_D$$

Only 3 operations are needed !

Other optimisations :

- Different scales :
 - Instead of changing samples scales, we change the filters scales.
- Normalisation of images computed from integral images and squared integral images :

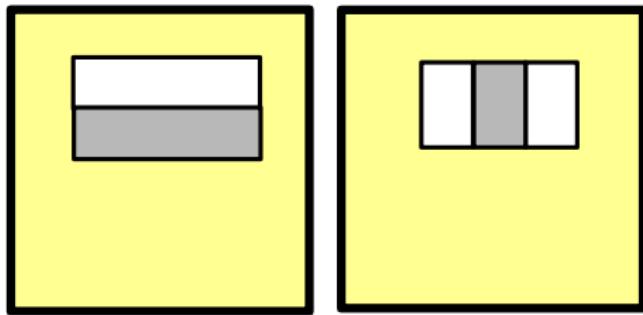
$$\sigma^2 = m^2 - \frac{1}{N} \sum x^2$$

Learning dataset :

- 5000 faces
 - front oriented
 - size 24x24 pixels
 - normalised
- 300 millions of non faces
 - taken from 9500 images without faces
- variability :
 - between individuals
 - illumination
 - pose

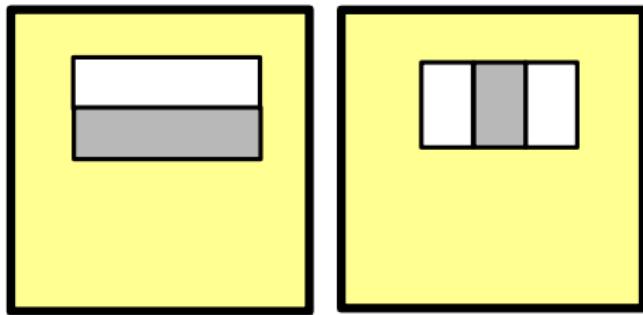


Selected filters by *boosting* : the first 2 features



- conduct to :
 - 100% detection rate (ratio true positives over positives)
 - 50% false positives rate (ratio false positives over negatives)
- able to eliminate non faces very fastly

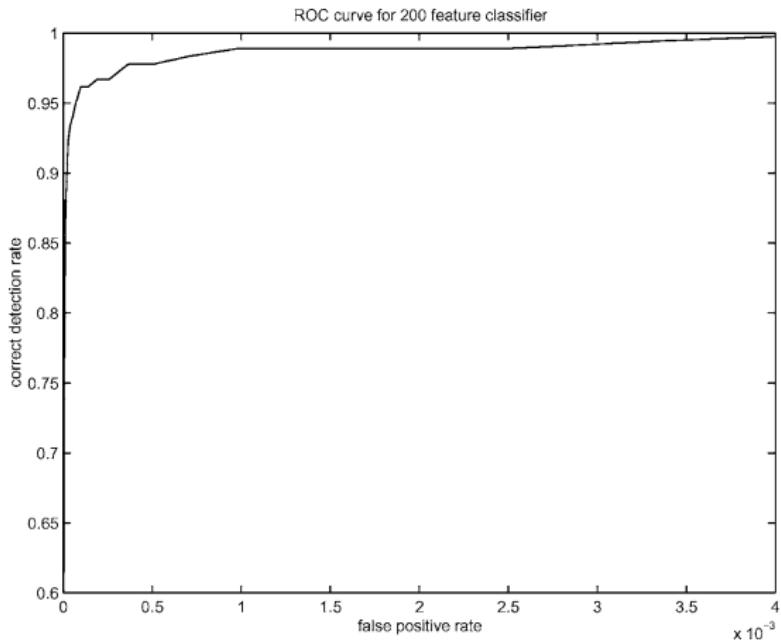
Selected filters by *boosting* : the first 2 features



- conduct to :
 - 100% detection rate (ratio true positives over positives)
 - 50% false positives rate (ratio false positives over negatives)
- able to eliminate non faces very fastly

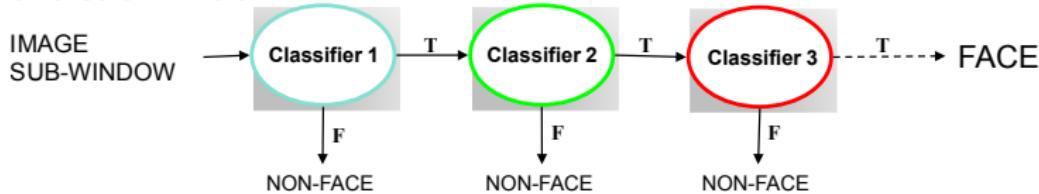
⇒ Fast elimination of non faces and increase of filters complexity

The 200 best filters selected : ROC curve

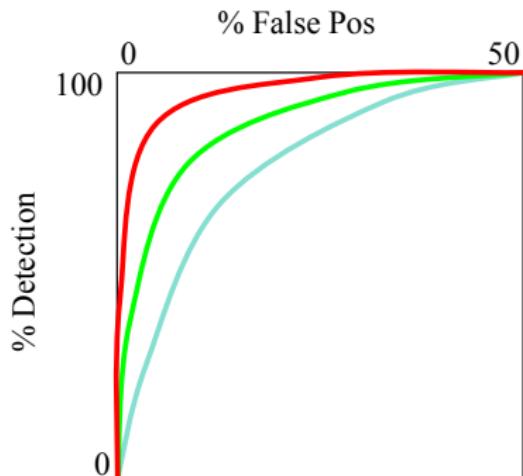


A 200-feature classifier can yield 95% detection rate and a false positive rate of 1 in 14084. To be practical for real application, the false positive rate must be closer to 1 in 1,000,000.

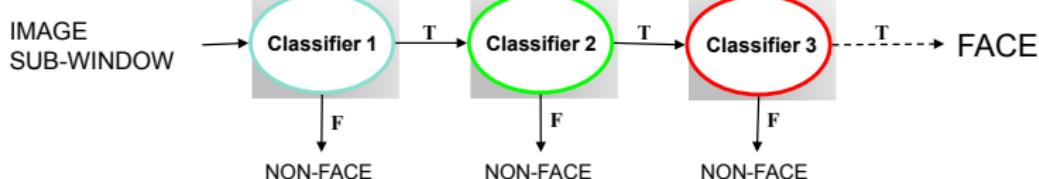
- We start with simple classifiers which reject many of the negative sub-windows while detecting almost all positive sub-windows
- Positive response from the first classifier triggers the evaluation of a second (more complex) classifier, and so on
- A negative outcome at any point leads to the immediate rejection of the sub-window



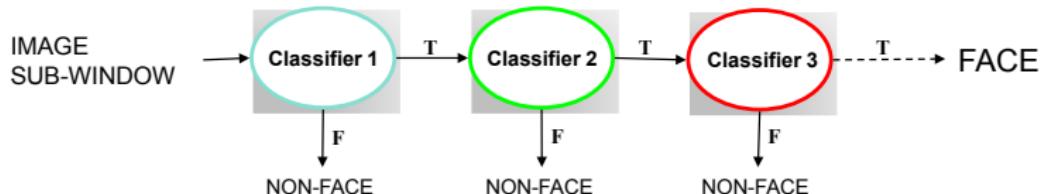
ROC Curve



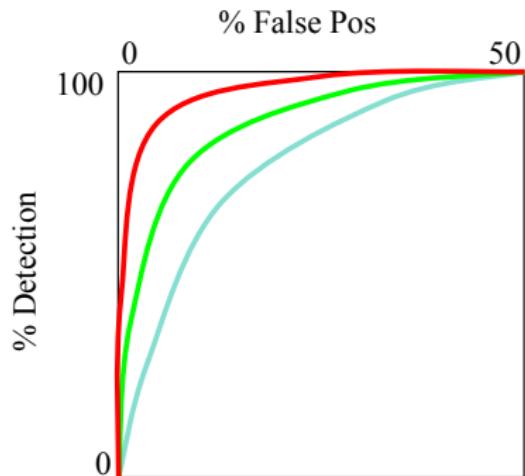
Chain classifiers that are progressively more complex and have lower false positive rates



Cascade structure



ROC Curve



$$F = \prod_{i=1}^K f_i$$

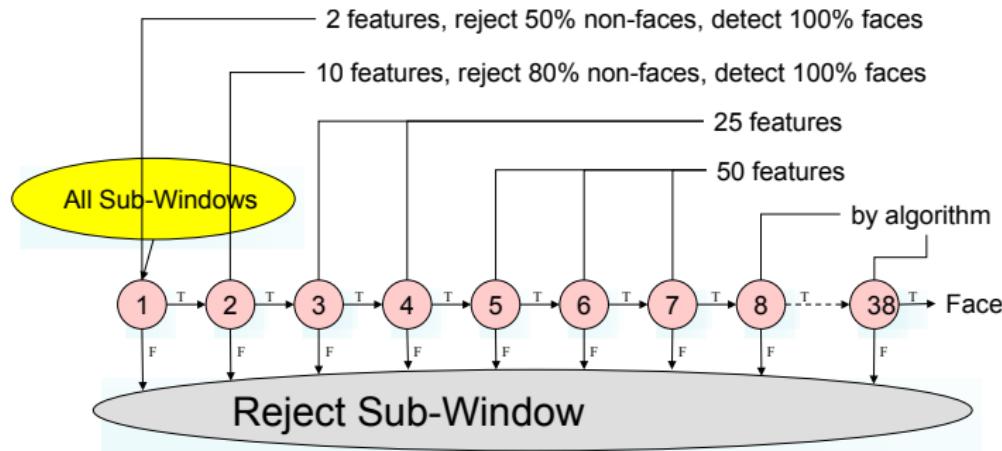
$$D = \prod_{i=1}^K d_i$$

A detection rate of 0.9 and a false positive rate on the order of 10^{-6} can be achieved by a 10-stage cascade if each stage has a detection rate of 0.99 ($0.99^{10} \approx 0.9$) and a false positive rate of about 0.30 ($0.3^{10} \approx 6 \cdot 10^{-6}$)

Algorithm for determining the stages of the cascade

- choose $f = \max_i f_i$ et $d = \min_i d_i$
- choose F
- P : set of faces
- N : set of non faces
- initialisations : $F_0 = 1.0$ $D_0 = 1.0$ $i = 0$
- while $F_i > F$
 - add a new stage : $i \leftarrow i + 1$
 - $n_i = 0$ $F_i = F_{i-1}$
 - while $F_i > f F_{i-1}$:
 - add filters : $n_i \leftarrow n_i + 1$
 - boosting with n_i filters
 - evaluation of cascade classifier on validation data : F_i and D_i
 - lower the threshold for having a detection rate at least : $d.D_{i-1}$
 - empty N
 - false positives go to N for the next step

The final cascade

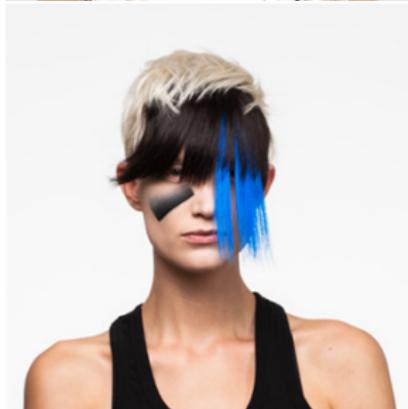
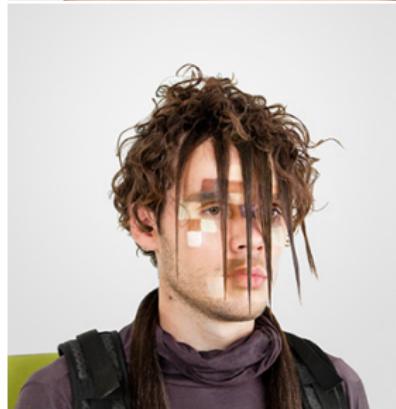
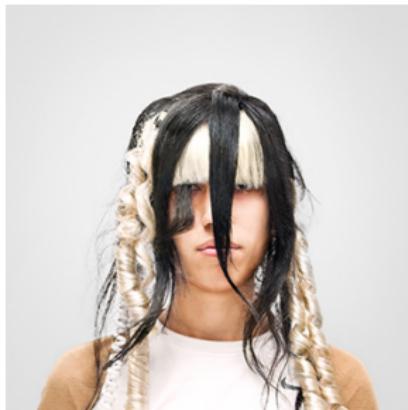


- Consists in combining in cascade more and more complex classifiers
 - 38 stages
 - total of 6060 classifiers
- Historical performances :
 - on a Pentium III, 700 MHz, processing of a 384x288 pixels images in 0.067 seconds (15 Hz)

- Other filters (rotation 45 deg)
- Nested cascades :
 - eyes detection
 - nose detection
 - smile detection

How to escape to facedetect ?

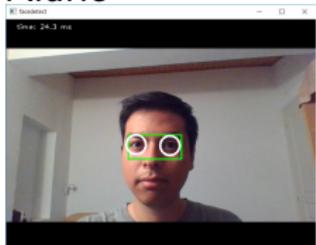
see <https://cvdazzle.com/>



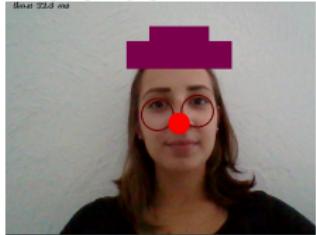
Let's play with facedetect !



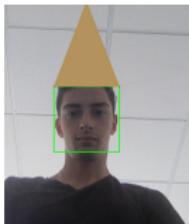
Aldric



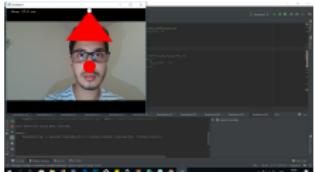
Alexander



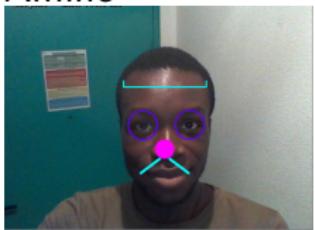
Alexia



Alexis



Amine



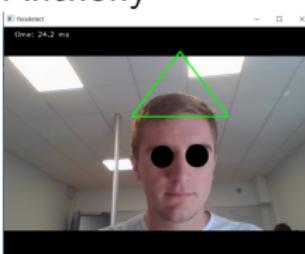
Amissan



Anass



Anthony



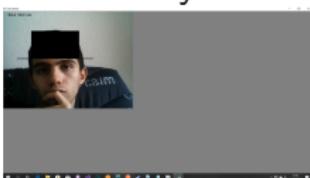
Armand



Augusto

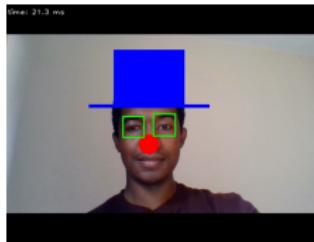


Barthelemy

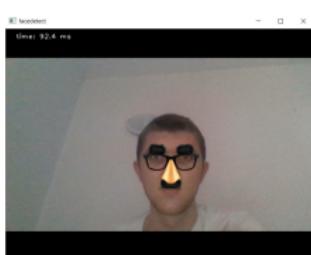


Bastien

Let's play with facedetect !



Betsara



Cyril



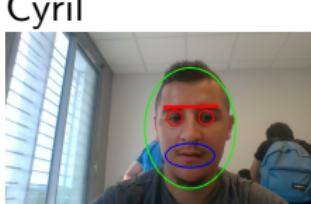
Erwan



Gabriel



Charles



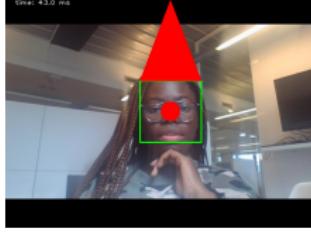
Damien



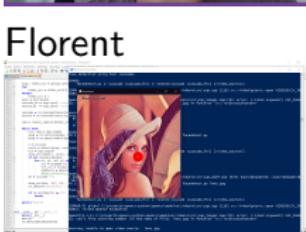
Florent



Come



Donelia



Florian



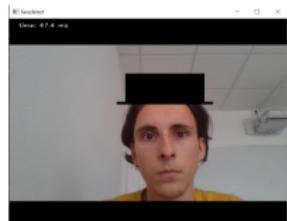
Guillaume

Hanting

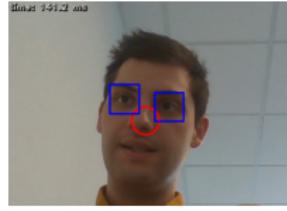
Let's play with facedetect !



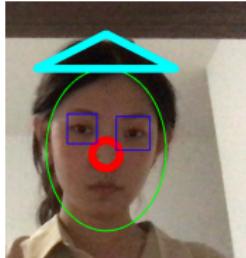
Ilyas



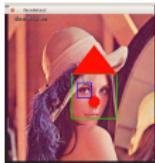
Jacques



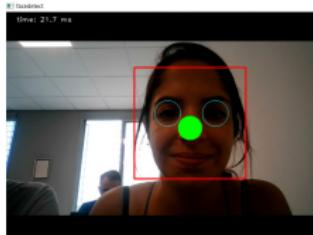
Jason



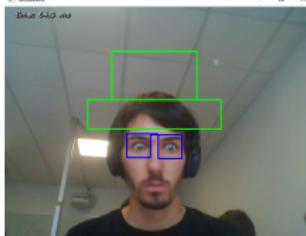
Jiazen



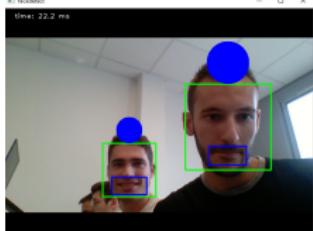
Loic



Meriam



Mael



Julien



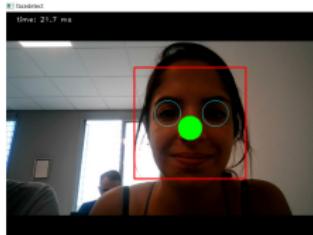
Maxime



Nidhal

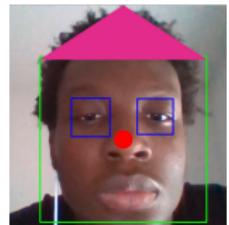


Kejia

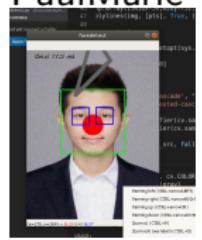


Paul

Let's play with facedetect !



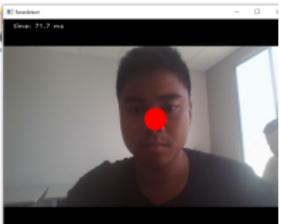
PaulMarie



Peidong



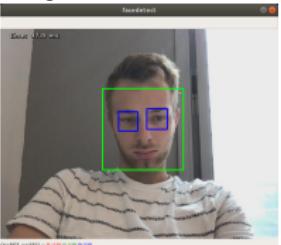
Pierre



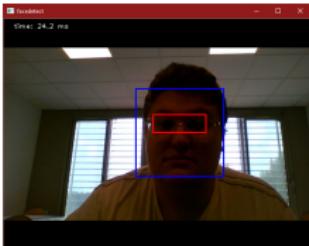
Quang



Rayan



Rémy



Richard



Robin



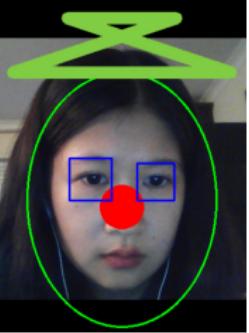
Romain



Ruben

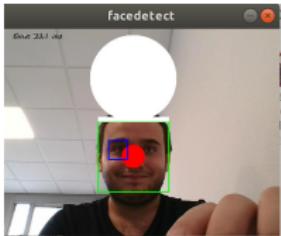


Sacha



Shuang

Let's play with facedetect !



Stéphane



Sylvain



Théo



Thomas



Tom



Vincent



Yan



Yao

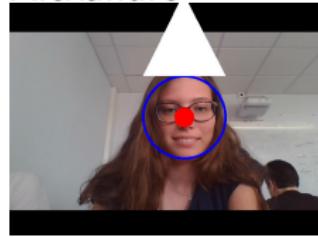


Yury

Let's play with facedetect !



Alexandre



Alexanne



Alis



Alexis G.



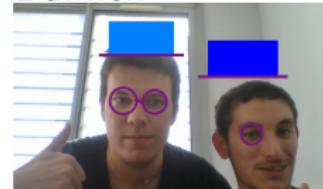
Damien



Clement



Damien

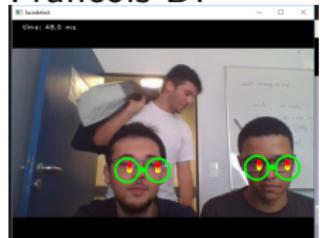


Denys

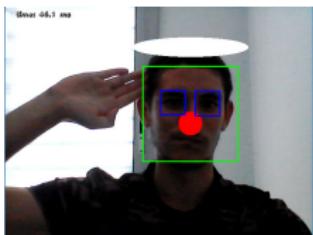
Let's play with facedetect !



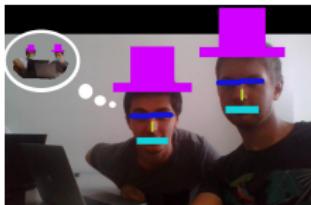
Francois D.



Gabriel



Guillaume



Jeremi



Kienan



Laura

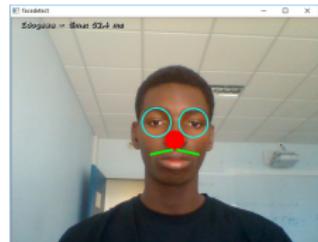


Mariana



Mathieu

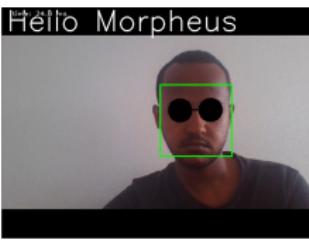
Let's play with facedetect !



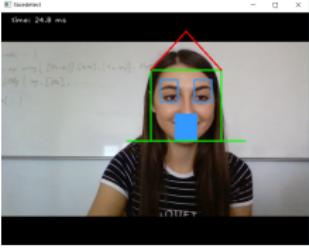
Mawuyram



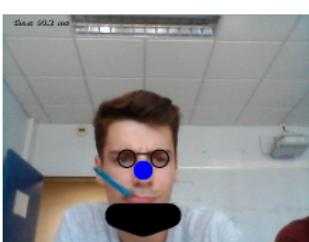
Maxime



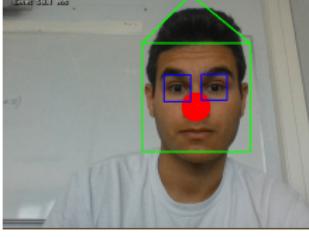
Natan



Olivia



Romain



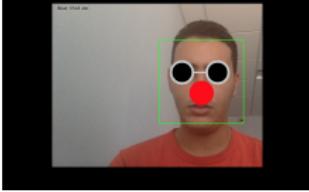
Sami



Sebastion

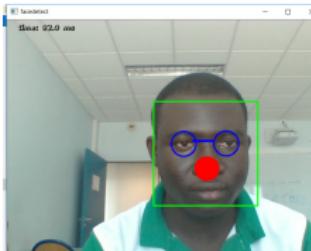


Simon

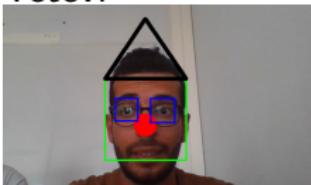


Soufiane

Jouons avec facedetect !



Tetevi



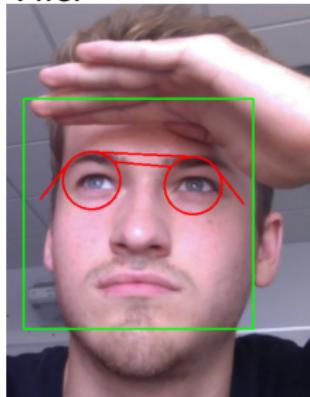
Walid



Wei



Yilei



Yon

AdaNet : <https://github.com/tensorflow/adanet>