Boosting:
combining elementary classifiers to learn a “strong” classifier

Pr. Fabien MOUTARDE
Center for Robotics
Mines Paris
PSL Université

Fabien.Moutarde@minesparis.psl.eu
http://people.minesparis.psl.eu/fabien.moutarde

Essential principle: ”wisdom of the crowd”

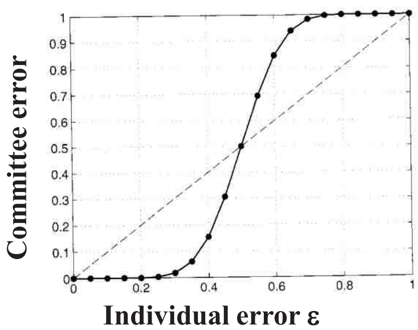
Set-up a “committee of experts”
each one can be wrong, but combining opinions increases the chance to obtain correct prediction!

Theoretical justification:

– suppose N *independent* classifiers, each with same error rate $E_{gen} = \varepsilon$

– decision by a “majority” vote is wrong if and only if more than half of the committee is wrong

$$\rightarrow \quad Error_{committee} = \sum_{k=N/2}^{N} C_k^N \varepsilon^k (1-\varepsilon)^{N-k}$$



Spectacular improvement of decision (under condition that ε<0.5!!)… …and the larger N (# of experts), the bigger the improvement

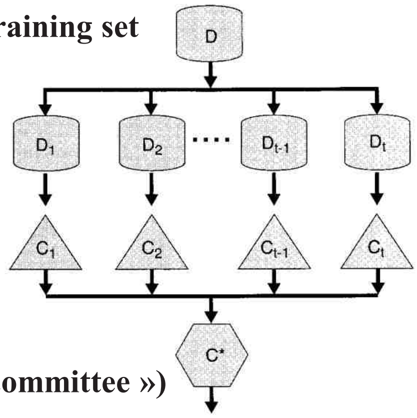
Various methods to produce the elementary classifiers to combine

- Use totally different algorithms
- Same algorithm, but with different parameters and/or initializations
- **<u>Modify the training set</u>**

Initial training set

Variants of the initial dataset (random sampling, different weightings, …)

Elementary classifiers obtained by same procedure applied to the variants of dataset



Global classifier (« committee »)

→ Very GENERAL methods, applicable to enhance any « elementary » algorithm

Bagging

Variants of training set obtained by random sampling (with re-placement) from initial dataset

(kind of "***<u>bootstrap</u>***"→ random duplication/erasure of some examples, depending on the variant)

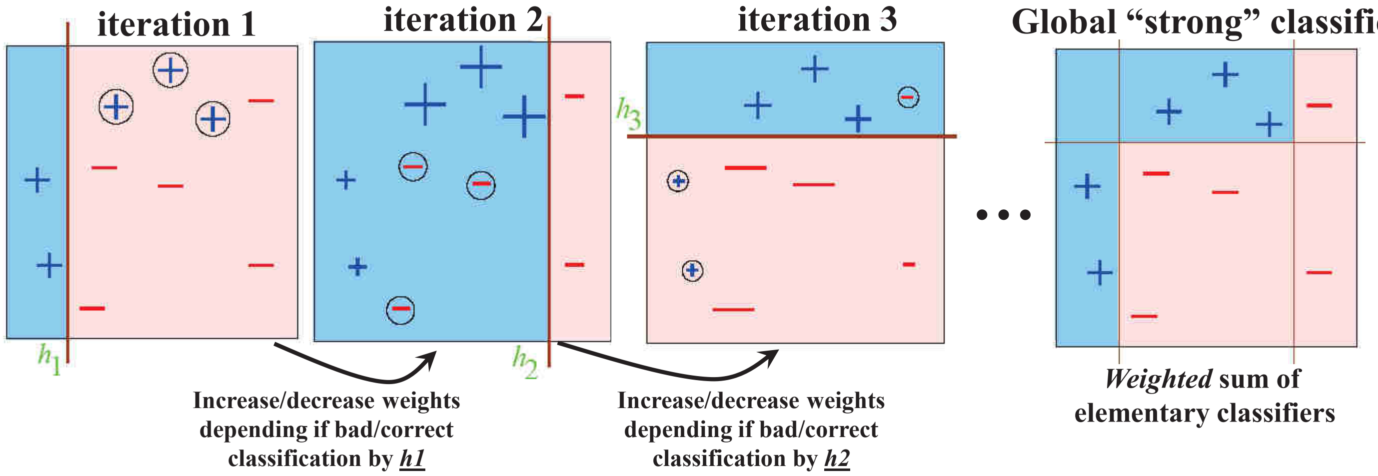
- **Useful and efficient in particular if the "elementary" algorithm is "sensitive to data noise" (because then different variants of training set shall induce quite different classifiers)**
- ***Reduces over-fitting*, because the final classifier is a kind of average of classifiers learnt on different realizations of the same data**

Boosting

<u>*Iterative* method for adding new classifiers to the committee:</u>

variants of training dataset obtained by *successive weightings* of the same examples

(computed for "focusing" on hard examples,
i.e. incorrectly classified by previous elementary classifiers)



adaBoost algorithm

adaBoost ("adaptive Boosting")

- **Initial training set:**
  $S=\{(x_1,u_1), \ ... \ ,(x_k,u_k)\}$, with $u_i \in \{+1,-1\}$, $i=1,k$
- **Initial weights:** $w_0(x_i)= 1/m$ for all $i=1,k$ (or $1/2p$ for pos, and $1/2n$ for neg)
- **For each iteration (or round) t from 1 to T, do:**
  1. Learn/choose 1 classification rule h_t on (S,w_t) using algorithm A
  2. Compute <u>*weighted* error</u> ε_t of h_t on (S,w_t) : $\varepsilon_t = \sum_{i=1}^{k} w_t(x_i)\times\|h_t(x_i)-u_i\|$
  3. Deduce <u>reliability score</u> α_t of h_t: $\alpha_t = \frac{1}{2}\ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$ *[α_t>0 if ε_t<0.5, and $\rightarrow+\infty$ if $\varepsilon_t\rightarrow 0$]*
  4. <u>Modify weights of examples</u>, i.e. for i from 1 to k, do:

$$w_{t+1}(x_i)=\frac{w_t(x_i)}{Z_t}\times\begin{cases} e^{-\alpha_t} \ si \ h_t(x_i)=u_i \ (i.e. \ x_i \ bien \ classé) \\ e^{+\alpha_t} \ si \ h_t(x_i)\neq u_i \ (i.e. \ x_i \ mal \ classé) \end{cases}$$

- **Output the global "strong" classifier:** $H(x)=signe\left(\sum_{t=1}^{T}\alpha_t h_t(x)\right)$

Convergence Theorem

Freund & Schapire (inventors of the algorithm) have demonstrated the following theorem:

If each elementary classifier has error-rate <0.5, then empirical error of H_T on S decreases *exponentially* with the number T of iterations

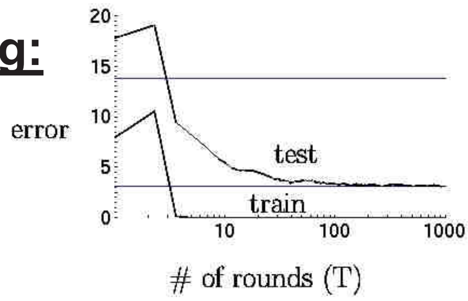
More precisely $E_{emp}(H_T) = \frac{1}{k}\sum_{i=1}^{k}\|H_T(x_i) - u_i\|$ **is bounded by:**

$$E_{emp}(H_T) \leq \prod_{t=1}^{T}\left[2\sqrt{\varepsilon_t(1-\varepsilon_t)}\right] = \prod_{t=1}^{T}\sqrt{1-4\gamma_t^2}$$

(where $\gamma_t = 0{,}5 - \varepsilon_t$ is the improvement of h_t compared to random decision)

Boosting and margins

Typical error training curve for boosting:
the *generalization error continues to decrease many iterations after training error becomes zero!!*

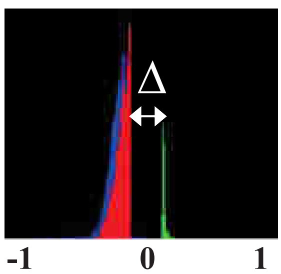


Reason: even after training_error reaches 0, adaBoost continues to increase *margins* i.e. output ≠ between negative and positive examples

Margin m of strong classifier H_T on example x_i:

$$m(H_T, x_i) = u_i \sum_{t=1}^{T}\alpha_t h_t(x_i) \Big/ \sum_{t=1}^{T}\alpha_t$$

$m(x_i) \in [-1;+1]$, and x_i correctly classified $\Leftrightarrow$ $m(x_i) > 0$, but *the more |m| increases, the larger the Δ separation between positive and negative examples*



-1 0 1

Risk of over-fitting by adaBoost?

- **Weight increase of « ambiguous » examples**
  ➔ risk of over-fitting?

- **Fortunately, generalization error bounded by:**

$$E_{gen}(H_T) < \Pr\left(m(H_T, x) \leq \theta\right) + O\left(\sqrt{\frac{\delta}{n\theta^2}}\right)$$

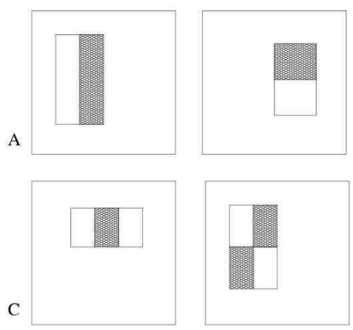
where n is the # of examples, and δ the VC-dimension of h_t family

→ if p($m(H_T,x)<\theta$) very low for a big-enough θ, then good generalization.

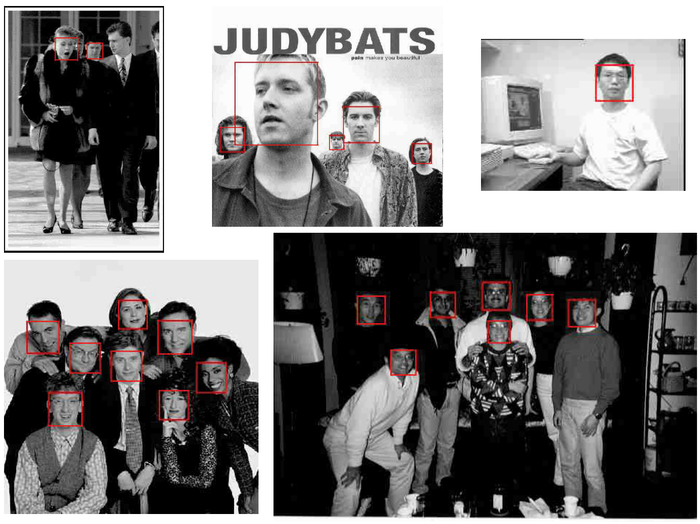
In practice the margin m increases with iterations, so this bound decreases ☺

adaBoost « Success story »

- **Visual object detection by selection-boosting of « Haar features »; initial example initial = face detection by Viola&Jones (2001)**



Weak classifiers = comparison of sums of pixels in adjacent rectangles



Result of applying strong classifier on multiple sub-windows of various sizes and positions ("window scanning")

Boosting as feature selection (and weighting)

adaBoost = weighted vote by a committee of *"weak classifiers"* obtained by iterative weightings of examples

➔ **Final STRONG classifier:** $H(x) = \mathrm{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$

Idea of Viola&Jones in 2001: ***use as weak classifier very simple boolean features selected in a family*** **(e.g. all Haar-like features)**
⇔ Weak Learner = search of feature with lowest weighted error

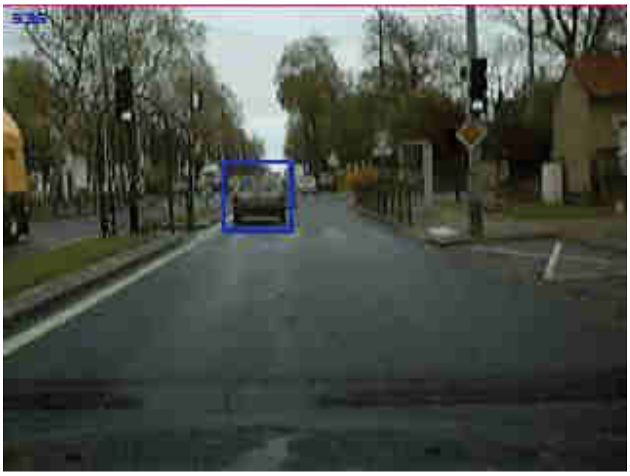
Using a 24x24 pixels detection window, with all possible combinations of horizontal&vertical location and scale of Haar, the full set of features has 45,396 ≠ features (and ~10 times more in a 32x32 window) ➔ brute-force exhaustive search possible!

Outcome of boosting with ≠ feature families

Typical connected-Control-Points selected during Adaboost training

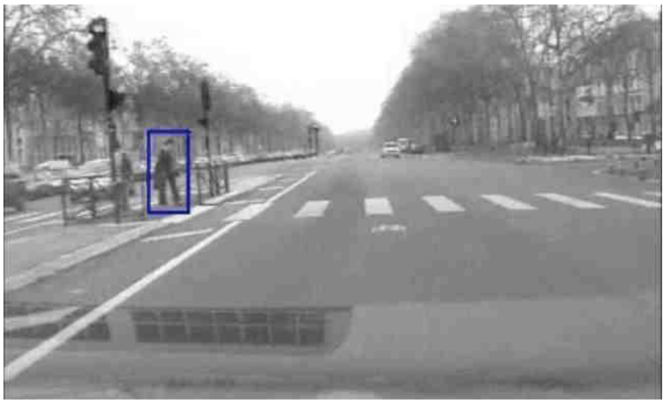
For comparison, typical Adaboost-selected Haar features

Result of car & pedestrian detection with boosting



Cars (from behind) : ~ 95% detection with < 1 false alarm / image

[Research conducted in ~2009 @ center for Robotics of MINES ParisTech]



Pedestrian (daytime) : ~80% detection with < 2 false alarms / image

Hyper-parameters for adaBoost

- Type of weak classifiers assembled
- The “weak learner” L which trains/generates a new weak classifier at each iteration (and potential hyper-parameters of L)
- # of iterations (= also the # of assembled weak classifiers)

Pros and cons of adaBoost

- **<u>Advantages</u>**
  - Can boost the performance of ANY learning algo (if able to handle weighting of examples)
  - Can build a strong classifier with ANY type of very weak classifiers (slightly better than random)
  - Can be used as an algorithm for selecting "weakly discriminative features" (cf. Viola & Jones)
- **<u>Drawbacks</u>**
  - Training time can be rather long (especially in "discriminative-feature selection" case)
  - Potential risk of over-fitting?