

These are the slides of the lecture

**Pattern Recognition**  
*Winter term 2020/21*  
*Friedrich-Alexander University of Erlangen-Nuremberg.*

These slides are released under Creative Commons License Attribution CC BY 4.0.

Please feel free to reuse any of the figures and slides, as long as you keep a reference to the source of these slides at <https://lme.tf.fau.de/teaching/> acknowledging the authors Niemann, Hornegger, Hahn, Steidl, Nöth, Seitz, Rodriguez, Das and Maier.

Erlangen, January 8, 2021  
Prof. Dr.-Ing. Andreas Maier

# Pattern Recognition (PR)

Prof. Dr.-Ing. Andreas Maier

Pattern Recognition Lab (CS 5), Friedrich-Alexander-Universität Erlangen-Nürnberg  
Winter Term 2020/21



# AdaBoost



## Boosting Methods

- Boosting is one of the most powerful learning techniques introduced in the last twenty years.
- Combines output of many *weak classifiers* to produce a powerful committee.
- The most popular boosting algorithm is called *AdaBoost* (Freund and Schapire, 1997).

## Boosting Methods (cont.)

### Definition

A weak classifier is one whose error rate is only slightly better than random guessing.

## Boosting Methods (cont.)

### Definition

A weak classifier is one whose error rate is only slightly better than random guessing.

Idea of boosting:

- Sequentially apply the *weak* classifier to repeatedly modified versions of the data.
- This produces a sequence of classifiers.
- Weighted majority vote yields the final prediction.

## Boosting Methods (cont.)

- Consider a two-class problem with  $y \in \{-1, +1\}$ .
- Given a set of observations  $\mathcal{D} = \{(\mathbf{x}_i, y_i); \quad i = 1, \dots, N\}$  and a classifier  $G(\mathbf{x})$ , the error rate on the training sample is:

$$\overline{\text{err}} = \frac{1}{N} \sum_{i=1}^N I(y_i \neq G(\mathbf{x}_i))$$

where  $I$  is the indicator function.

## Boosting Methods (cont.)

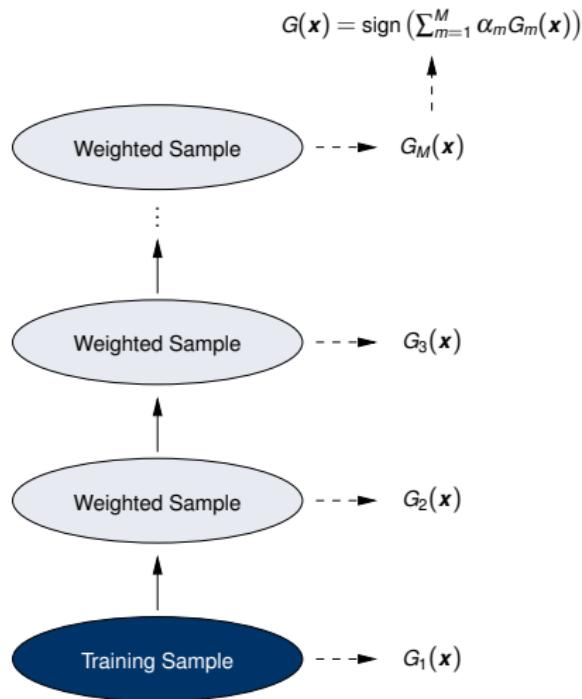
- Sequentially applied weak classifiers produce a sequence  $G_m(\mathbf{x})$  with  $m = 1, 2, \dots, M$
- The combined prediction is then:

$$G(\mathbf{x}) = \text{sign} \left( \sum_{m=1}^M \alpha_m G_m(\mathbf{x}) \right)$$

- The weighting factors  $\alpha_1, \dots, \alpha_M$  are computed by the boosting algorithm.
- Each  $\alpha_m$  weights the output of the corresponding classifier  $G_m(\mathbf{x})$ .

# Boosting Methods

Schematic of boosting procedure:



## Boosting Methods (cont.)

Modifications on the data:

- Each boosting step consists of applying weights  $w_1, w_2, \dots, w_N$  to the training samples.
- Initially, the weights are set to  $w_i = 1/N$ .
- Thus the first classifier in the sequence is trained the usual way.
- For  $m \geq 2$  the weights are individually modified.
- The classifiers  $G_m$ , with  $m \geq 2$  are trained on differently-weighted samples.

## Boosting Methods

Weighting scheme:

- At step  $m$  the misclassified observations from  $G_{m-1}$  have increased weights.
- The weights for correctly classified samples from  $G_{m-1}$  have decreased weights.

## Boosting Methods

Weighting scheme:

- At step  $m$  the misclassified observations from  $G_{m-1}$  have increased weights.
- The weights for correctly classified samples from  $G_{m-1}$  have decreased weights.

Effects of the weighting scheme:

- Observations that are difficult to classify correctly get ever-increasing influence.
- Each successive classifier is forced to concentrate more on those observations that were misclassified by the previous one.

# AdaBoost

AdaBoost algorithm:

	Initialize weights: $w_i \leftarrow 1/N, i = 1, \dots, N$
	$m \leftarrow 1$
	Fit classifier $G_m(\mathbf{x})$ to training data using $\mathbf{w}$
	Compute classification error: $\text{err}_m = \frac{\sum_{i=1}^N w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^N w_i}$
	Compute classifier weights: $\alpha_m = \log \left( \frac{1 - \text{err}_m}{\text{err}_m} \right)$
	Compute new sample weights: $w_i \leftarrow w_i \exp [\alpha_m I(y_i \neq G_m(x_i))], i = 1, \dots, N$
	$m \leftarrow m + 1$
	$m = M$
	Output: $G(\mathbf{x}) = \text{sign} \left( \sum_{m=1}^M \alpha_m G_m(\mathbf{x}) \right)$

## AdaBoost (cont.)

Notes:

- This version of AdaBoost is called **discrete**, because each  $G_m(\mathbf{x})$  returns a discrete class label.
- AdaBoost can be modified to return also a real-valued prediction in the interval  $[-1, +1]$ .
- Instead of taking just any classifier for  $G_m(\mathbf{x})$ , that classifier may be used that results in the smallest error at step  $m$ .

## AdaBoost (cont.)

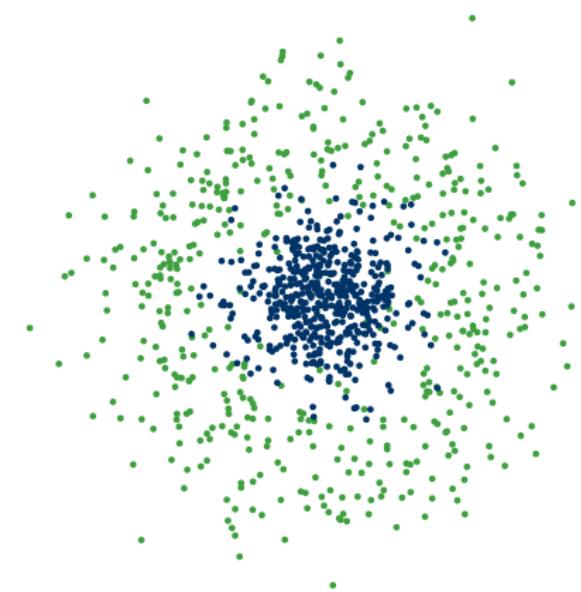
Notes:

- This version of AdaBoost is called **discrete**, because each  $G_m(\mathbf{x})$  returns a discrete class label.
- AdaBoost can be modified to return also a real-valued prediction in the interval  $[-1, +1]$ .
- Instead of taking just any classifier for  $G_m(\mathbf{x})$ , that classifier may be used that results in the smallest error at step  $m$ .

AdaBoost dramatically increases the performance of even a very weak classifier.

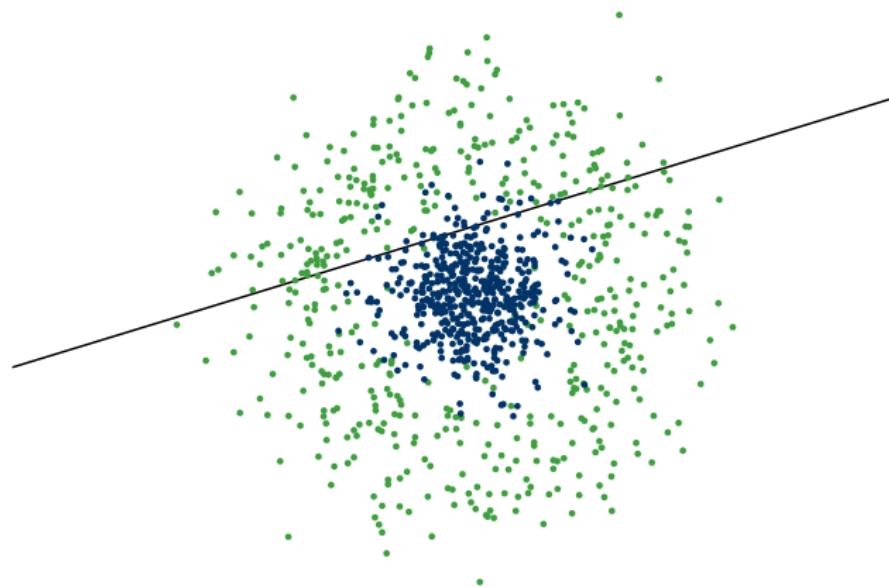
## AdaBoost (cont.)

Example from J. Matas and J. Šochman  
(Centre for Machine Perception, Technical University, Prague):



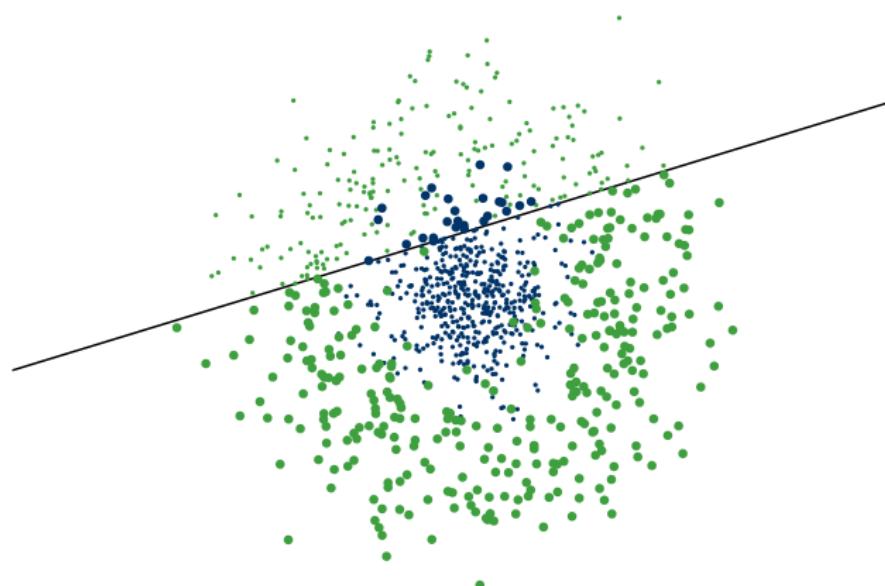
## AdaBoost (cont.)

Example from J. Matas and J. Šochman  
(Centre for Machine Perception, Technical University, Prague):



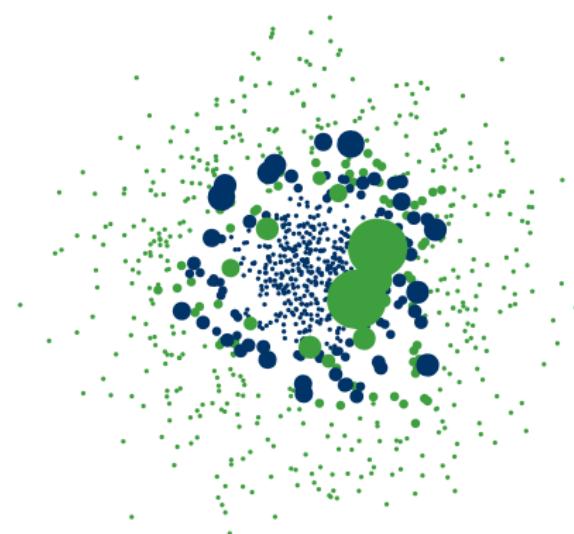
## AdaBoost (cont.)

Example from J. Matas and J. Šochman  
(Centre for Machine Perception, Technical University, Prague):



## AdaBoost (cont.)

Example from J. Matas and J. Šochman  
(Centre for Machine Perception, Technical University, Prague):



## AdaBoost (cont.)

Example from J. Matas and J. Šochman  
(Centre for Machine Perception, Technical University, Prague):

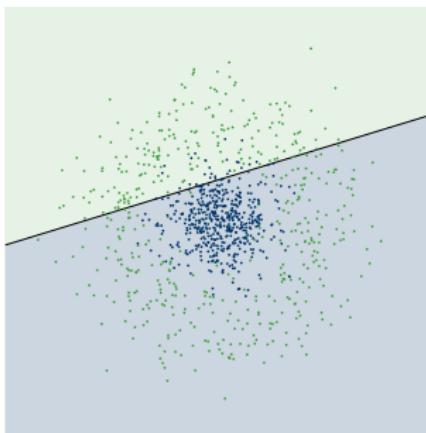


Fig.: Final classifier (based on a perceptron).

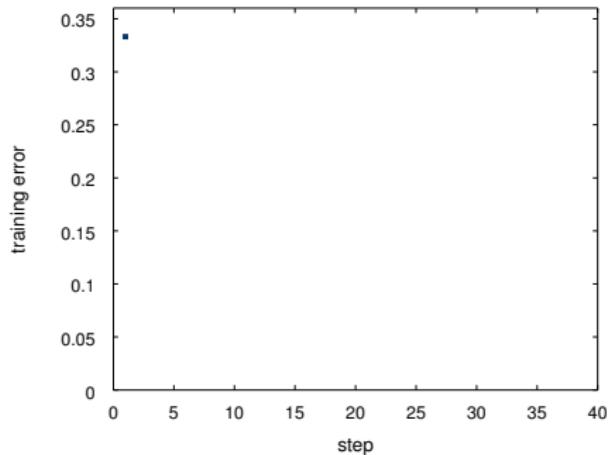


Fig.: Reduction of the classification error w. r. t. the sequence length.

## AdaBoost (cont.)

Example from J. Matas and J. Šochman  
(Centre for Machine Perception, Technical University, Prague):

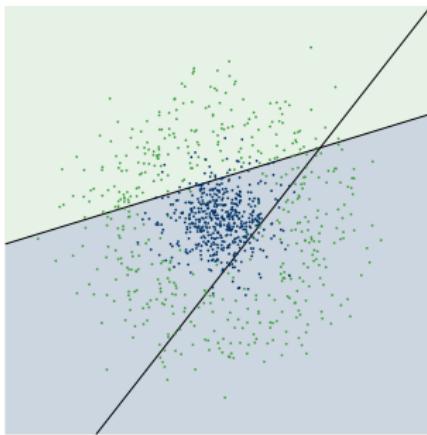


Fig.: Final classifier (based on a perceptron).

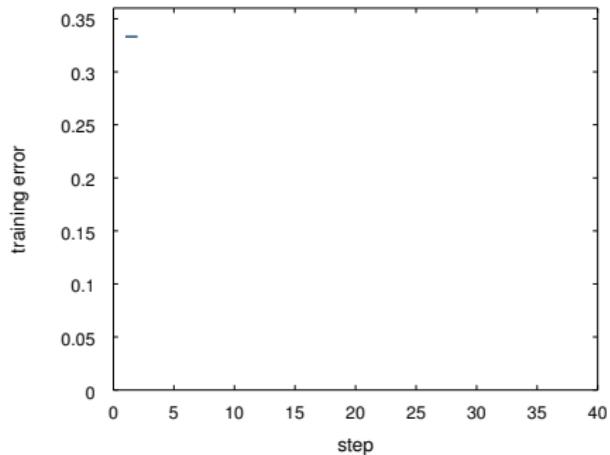


Fig.: Reduction of the classification error w. r. t. the sequence length.

## AdaBoost (cont.)

Example from J. Matas and J. Šochman  
(Centre for Machine Perception, Technical University, Prague):

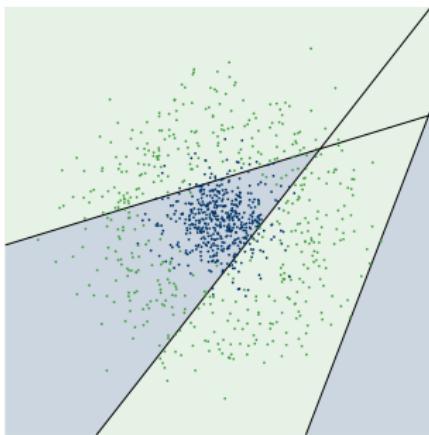


Fig.: Final classifier (based on a perceptron).

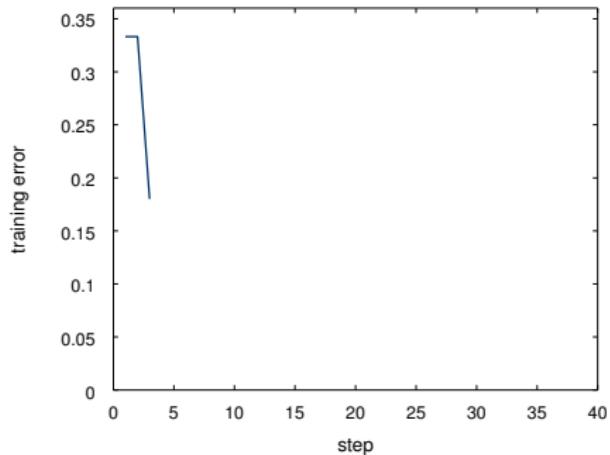


Fig.: Reduction of the classification error w. r. t. the sequence length.

## AdaBoost (cont.)

Example from J. Matas and J. Šochman  
(Centre for Machine Perception, Technical University, Prague):

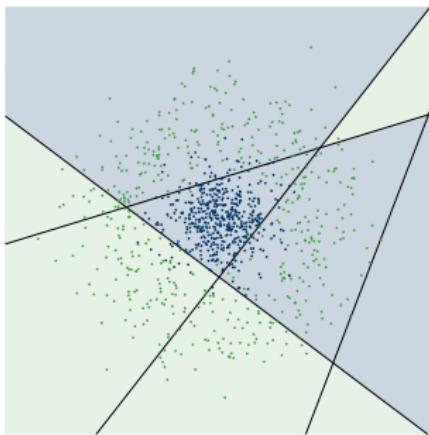


Fig.: Final classifier (based on a perceptron).

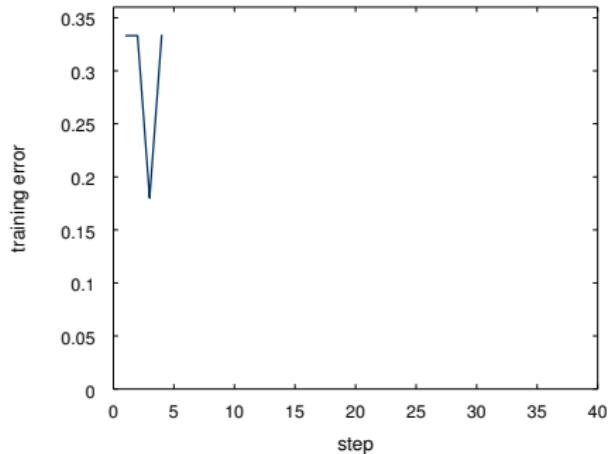


Fig.: Reduction of the classification error w. r. t. the sequence length.

## AdaBoost (cont.)

Example from J. Matas and J. Šochman  
(Centre for Machine Perception, Technical University, Prague):

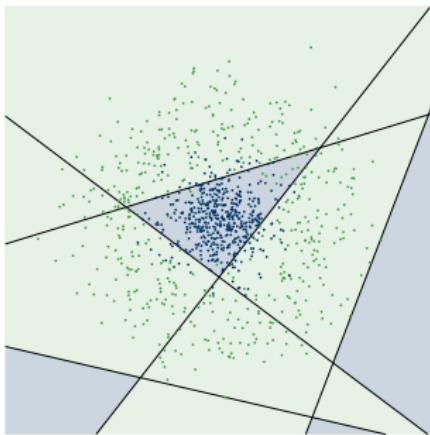


Fig.: Final classifier (based on a perceptron).

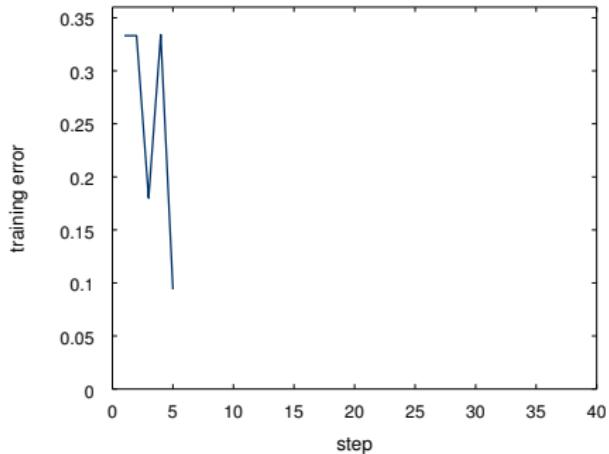


Fig.: Reduction of the classification error w. r. t. the sequence length.

## AdaBoost (cont.)

Example from J. Matas and J. Šochman  
(Centre for Machine Perception, Technical University, Prague):

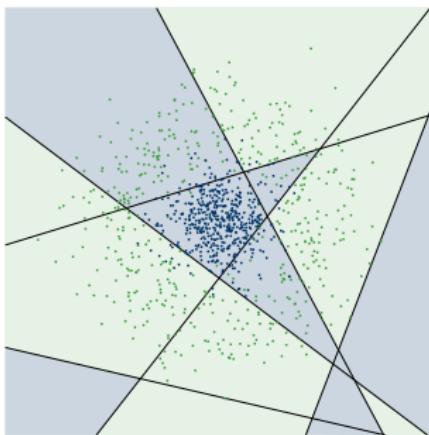


Fig.: Final classifier (based on a perceptron).

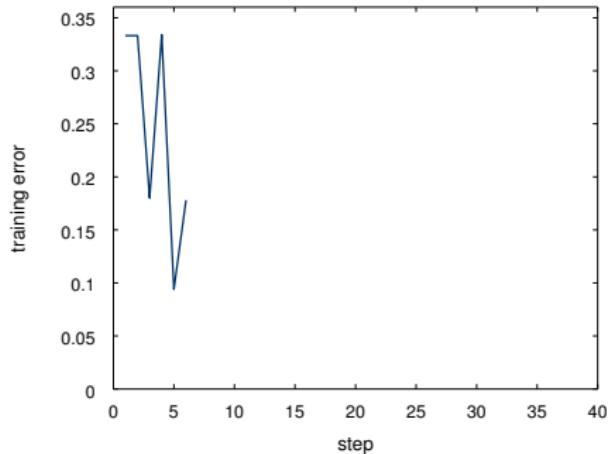


Fig.: Reduction of the classification error w. r. t. the sequence length.

## AdaBoost (cont.)

Example from J. Matas and J. Šochman  
(Centre for Machine Perception, Technical University, Prague):

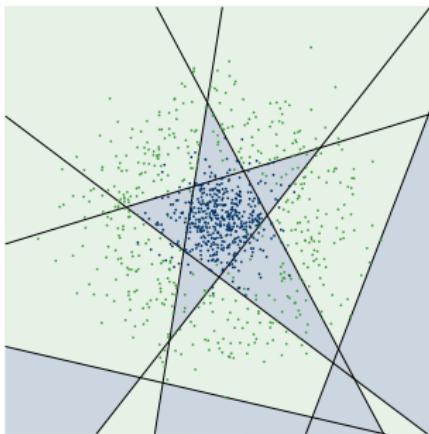


Fig.: Final classifier (based on a perceptron).

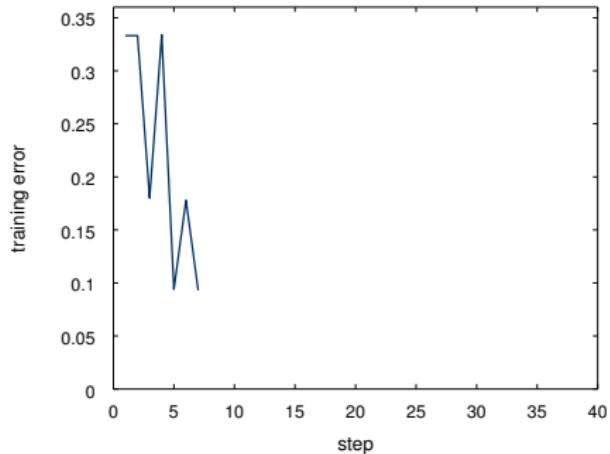


Fig.: Reduction of the classification error w. r. t. the sequence length.

## AdaBoost (cont.)

Example from J. Matas and J. Šochman  
(Centre for Machine Perception, Technical University, Prague):

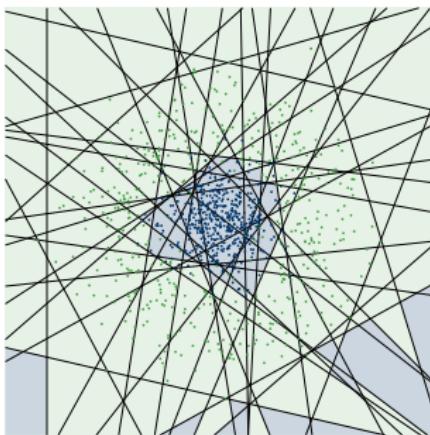


Fig.: Final classifier (based on a perceptron).

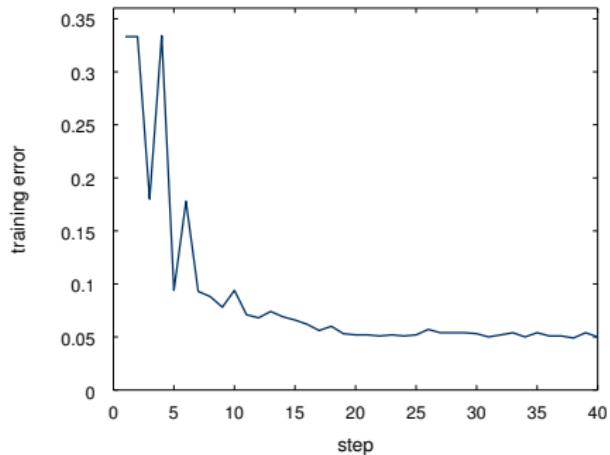


Fig.: Reduction of the classification error w. r. t. the sequence length.



**Pattern  
Recognition  
Lab**



FRIEDRICH-ALEXANDER  
UNIVERSITÄT  
ERLANGEN-NÜRNBERG  
TECHNISCHE FAKULTÄT

**Next Time in**

# **Pattern Recognition**



## Exponential Loss

- Boosting fits an additive model in a set of elementary basis functions:

$$f_M(\mathbf{x}) = \sum_{m=1}^M \beta_m b(\mathbf{x}; \gamma_m)$$

where  $\beta_m$  are expansion coefficients and  $b(\mathbf{x}; \gamma_m)$  is a basis function given a set of parameters  $\gamma_m$ .

## Exponential Loss

- Boosting fits an additive model in a set of elementary basis functions:

$$f_M(\mathbf{x}) = \sum_{m=1}^M \beta_m b(\mathbf{x}; \gamma_m)$$

where  $\beta_m$  are expansion coefficients and  $b(\mathbf{x}; \gamma_m)$  is a basis function given a set of parameters  $\gamma_m$ .

- Additive expansions are very popular in learning techniques:
  - single-hidden-layer neural networks (perceptron)
  - wavelets
  - classification trees
  - etc.

## Exponential Loss (cont.)

- Expansion models are typically fit by minimizing a loss function  $L$  averaged over the training data:

$$\min_{\{\beta_m, \gamma_m\}_1^M} \left\{ \sum_{i=1}^N L(y_i, f_M(\mathbf{x}_i)) = \sum_{i=1}^N L \left( y_i, \sum_{m=1}^M \beta_m b(\mathbf{x}_i; \gamma_m) \right) \right\} \quad (1)$$

## Exponential Loss (cont.)

- Expansion models are typically fit by minimizing a loss function  $L$  averaged over the training data:

$$\min_{\{\beta_m, \gamma_m\}_1^M} \left\{ \sum_{i=1}^N L(y_i, f_M(\mathbf{x}_i)) = \sum_{i=1}^N L \left( y_i, \sum_{m=1}^M \beta_m b(\mathbf{x}_i; \gamma_m) \right) \right\} \quad (1)$$

- Forward stagewise modeling approximates the solution to (1):
  - New basis functions are sequentially added.
  - Parameters and coefficients of already added functions are not changed.
  - At each iteration, the subproblem of fitting just a single basis function is solved.

## Exponential Loss (cont.)

- The  $m$ -th subproblem may be rewritten as:

$$(\beta_m, \gamma_m) = \underset{\beta, \gamma}{\operatorname{argmin}} \sum_{i=1}^N L(y_i, f_{m-1}(\mathbf{x}_i) + \beta b(\mathbf{x}_i; \gamma))$$

## Exponential Loss (cont.)

- The  $m$ -th subproblem may be rewritten as:

$$(\beta_m, \gamma_m) = \underset{\beta, \gamma}{\operatorname{argmin}} \sum_{i=1}^N L(y_i, f_{m-1}(\mathbf{x}_i) + \beta b(\mathbf{x}_i; \gamma))$$

- AdaBoost is equivalent to a forward stagewise additive modeling using an exponential loss function:

$$L(y, f(\mathbf{x})) = \exp(-y f(\mathbf{x}))$$

## Exponential Loss (cont.)

Proof:

- For AdaBoost, the basis functions are the classifiers  $G_m(\mathbf{x}) \in \{-1, +1\}$ .
- Using the **exponential loss function**, one must solve at each step:

$$(\beta_m, G_m) = \underset{\beta, G}{\operatorname{argmin}} \sum_{i=1}^N \exp \left[ -y_i (f_{m-1}(\mathbf{x}_i) + \beta G(\mathbf{x}_i)) \right]$$

- Using the weight  $w_i^{(m)} = \exp(-y_i f_{m-1}(\mathbf{x}_i))$  this can be rewritten as:

$$(\beta_m, G_m) = \underset{\beta, G}{\operatorname{argmin}} \sum_{i=1}^N w_i^{(m)} \exp(-\beta y_i G(\mathbf{x}_i))$$

## Exponential Loss (cont.)

Observations:

- Since  $w_i^{(m)}$  is independent of  $\beta$  and  $G(\mathbf{x})$ , it can be seen as a weight applied to each observation.
- However, this weight depends on  $f_{m-1}$ , so the weights change with each iteration  $m$ .

## Exponential Loss (cont.)

$$\begin{aligned}
 (\beta_m, G_m) &= \operatorname{argmin}_{\beta, G} \sum_{i=1}^N w_i^{(m)} \exp(-\beta y_i G(\mathbf{x}_i)) \\
 &= \operatorname{argmin}_{\beta, G} e^{-\beta} \sum_{y_i=G(\mathbf{x}_i)} w_i^{(m)} + e^{\beta} \sum_{y_i \neq G(\mathbf{x}_i)} w_i^{(m)} \\
 &= \operatorname{argmin}_{\beta, G} \left( e^{\beta} - e^{-\beta} \right) \sum_{i=1}^N w_i^{(m)} I(y_i \neq G(\mathbf{x}_i)) + e^{-\beta} \sum_{i=1}^N w_i^{(m)}
 \end{aligned}$$

## Exponential Loss (cont.)

$$\begin{aligned}
 (\beta_m, G_m) &= \underset{\beta, G}{\operatorname{argmin}} \sum_{i=1}^N w_i^{(m)} \exp(-\beta y_i G(\mathbf{x}_i)) \\
 &= \underset{\beta, G}{\operatorname{argmin}} e^{-\beta} \sum_{y_i=G(\mathbf{x}_i)} w_i^{(m)} + e^{\beta} \sum_{y_i \neq G(\mathbf{x}_i)} w_i^{(m)} \\
 &= \underset{\beta, G}{\operatorname{argmin}} \left( e^{\beta} - e^{-\beta} \right) \sum_{i=1}^N w_i^{(m)} I(y_i \neq G(\mathbf{x}_i)) + e^{-\beta} \sum_{i=1}^N w_i^{(m)}
 \end{aligned}$$

- Thus, for any value  $\beta > 0$  the solution for  $G_m(\mathbf{x})$  is:

$$G_m = \underset{G}{\operatorname{argmin}} \sum_{i=1}^N w_i^{(m)} I(y_i \neq G(\mathbf{x}_i))$$

## Exponential Loss (cont.)

- Plugging the reformulated  $G_m(\mathbf{x})$  into the objective function and solving for  $\beta_m$  yields:

$$\beta_m = \frac{1}{2} \log \frac{1 - \text{err}_m}{\text{err}_m}$$

with  $\text{err}_m$  being the minimized weighted error rate:

$$\text{err}_m = \frac{\sum_{i=1}^N w_i^{(m)} I(y_i \neq G_m(\mathbf{x}_i))}{\sum_{i=1}^N w_i^{(m)}}$$

## Exponential Loss (cont.)

- From the update formula of the approximation:

$$f_m(\mathbf{x}) = f_{m-1}(\mathbf{x}) + \beta_m G_m(\mathbf{x})$$

we can calculate the weights for the next iteration:

$$\begin{aligned} w_i^{(m+1)} &= w_i^{(m)} e^{-\beta_m y_i G_m(\mathbf{x}_i)} \\ &\quad (\text{using: } -y_i G_m(\mathbf{x}_i) = 2I(y_i \neq G_m(\mathbf{x}_i)) - 1) \\ &= w_i^{(m)} e^{\alpha_m I(y_i \neq G_m(\mathbf{x}_i))} e^{-\beta_m} \end{aligned}$$

with  $\alpha_m = 2\beta_m$ .

□

## Exponential Loss (cont.)

Compare result to initial AdaBoost algorithm:

- Exponential loss:

$$\begin{aligned}\beta_m &= \frac{1}{2} \log \frac{1 - \text{err}_m}{\text{err}_m} \\ \alpha_m &= 2\beta_m \\ w_i^{(m+1)} &= w_i^{(m)} e^{\alpha_m I(y_i \neq G_m(\mathbf{x}_i))} e^{-\beta_m}\end{aligned}$$

## Exponential Loss (cont.)

Compare result to initial AdaBoost algorithm:

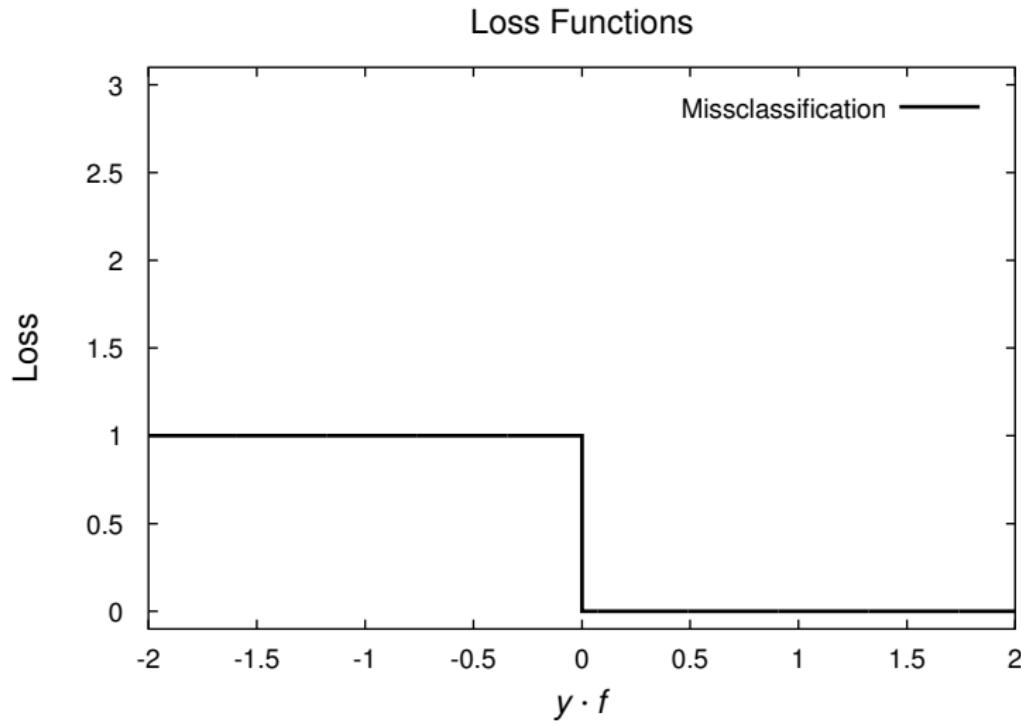
- Exponential loss:

$$\begin{aligned}\beta_m &= \frac{1}{2} \log \frac{1 - \text{err}_m}{\text{err}_m} \\ \alpha_m &= 2\beta_m \\ w_i^{(m+1)} &= w_i^{(m)} e^{\alpha_m I(y_i \neq G_m(\mathbf{x}_i))} e^{-\beta_m}\end{aligned}$$

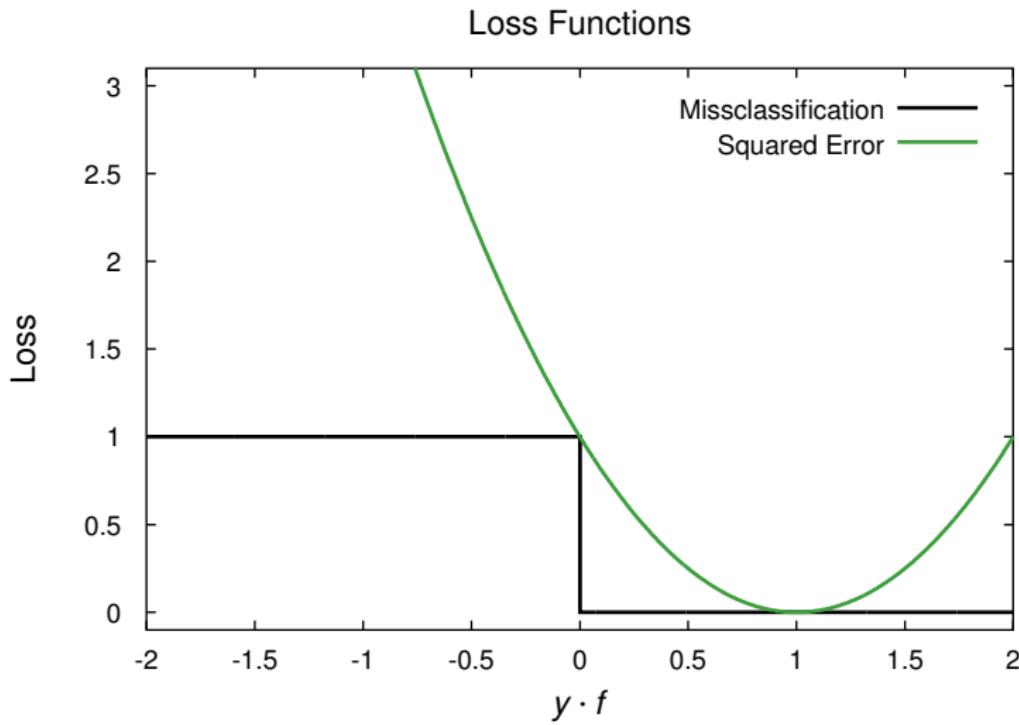
- AdaBoost:

$$\begin{aligned}\alpha_m &= \log \frac{1 - \text{err}_m}{\text{err}_m} \\ w_i &\leftarrow w_i e^{\alpha_m I(y_i \neq G_m(\mathbf{x}_i))}\end{aligned}$$

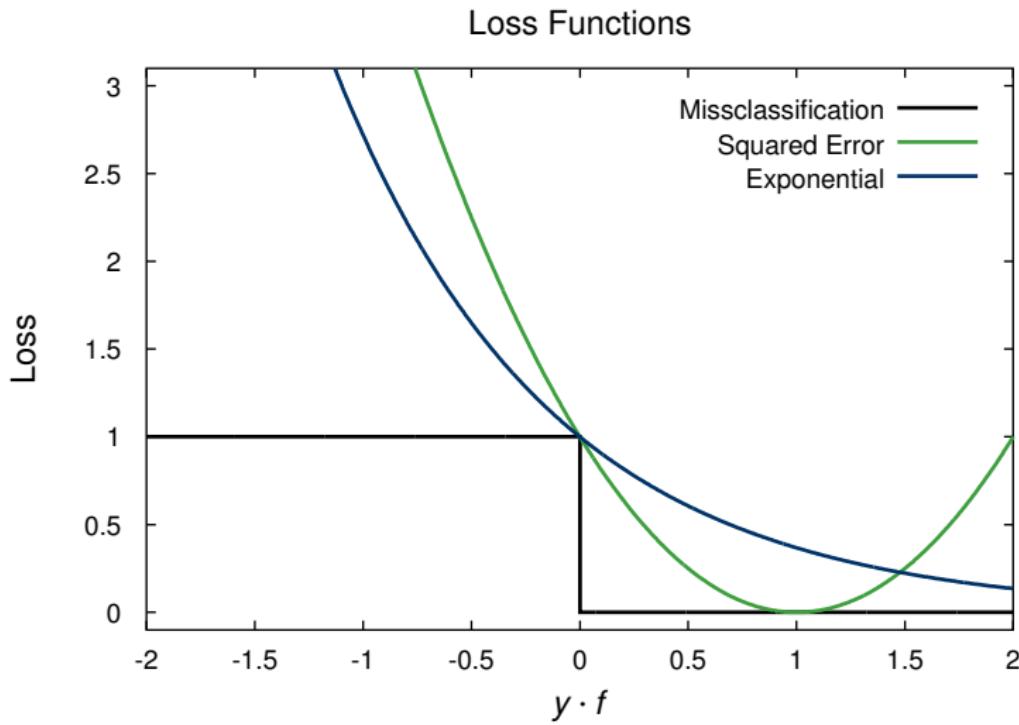
## Exponential Loss (cont.)



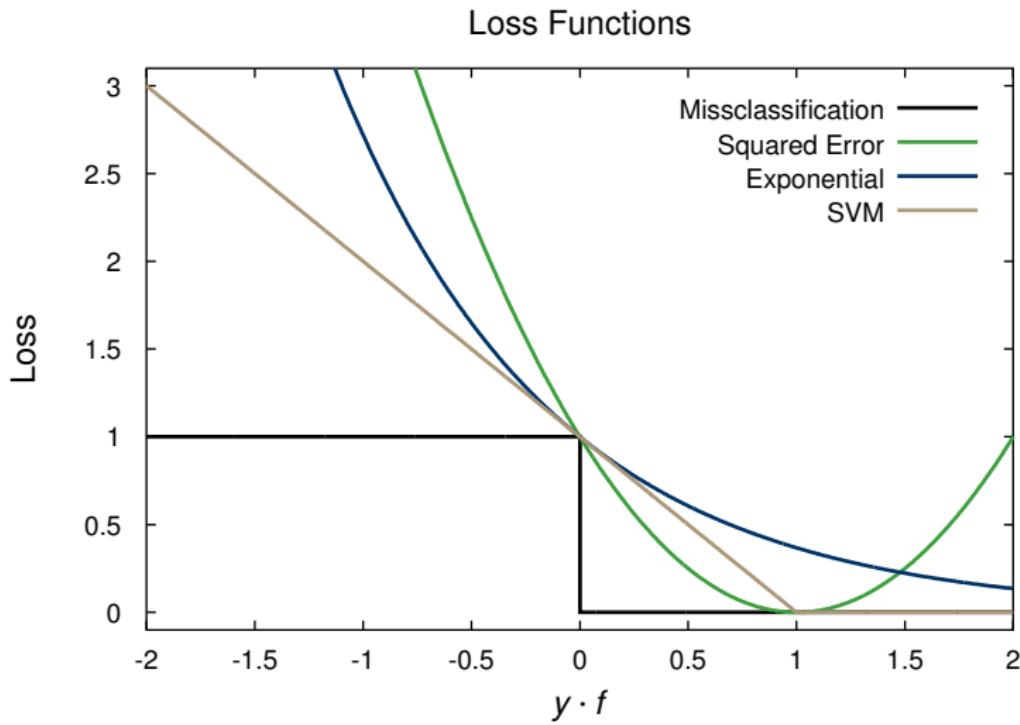
## Exponential Loss (cont.)



## Exponential Loss (cont.)



## Exponential Loss (cont.)



## Exponential Loss (cont.)

Discussion:

- The equivalence of AdaBoost to forward stagewise additive modeling was discovered five years after its invention.
- The AdaBoost criterion yields a monotone decreasing function of the margin:  $y f(\mathbf{x})$ .
- In  $\{-1, 1\}$  classification, the margin plays a role similar to the residuals  $y - f(\mathbf{x})$  in regression:
  - Observations with  $y_i f(\mathbf{x}_i) > 0$  are classified correctly
  - Observations with  $y_i f(\mathbf{x}_i) < 0$  are misclassified
  - The decision boundary is at  $f(\mathbf{x}) = 0$

## Exponential Loss (cont.)

Discussion (cont.):

- The goal of the classification algorithm is to produce positive margins as frequently as possible.
- Thus, any loss criterion should penalize negative margins more heavily than positive ones!
- The exponential criterion concentrates much more influence on observations with large negative margins.

## Exponential Loss (cont.)

Discussion (cont.):

- The goal of the classification algorithm is to produce positive margins as frequently as possible.
- Thus, any loss criterion should penalize negative margins more heavily than positive ones!
- The exponential criterion concentrates much more influence on observations with large negative margins.

Due to the exponential loss, AdaBoost performance is known to degrade rapidly:

- in situations of noisy data
- when there are wrong class labels in the training data



**Pattern  
Recognition  
Lab**



FRIEDRICH-ALEXANDER  
UNIVERSITÄT  
ERLANGEN-NÜRNBERG

TECHNISCHE FAKULTÄT

**Next Time in**

# **Pattern Recognition**



# Face Detection

Famous algorithm developed by [Viola and Jones](#) in 2001.

Contributions:

- Integral images for feature computation.
- Usage of AdaBoost for boosting.
- Classifier cascade for fast rejection of non-face regions.

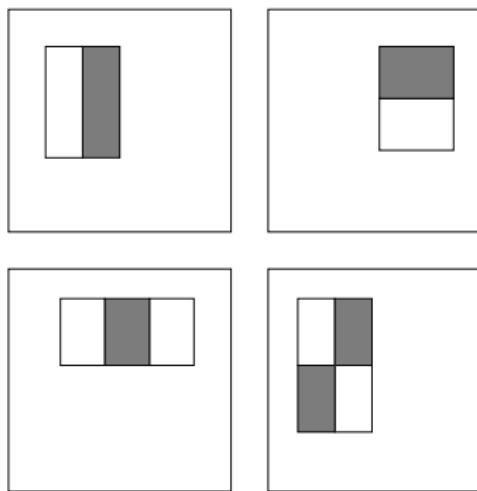
Various implementations available: for example look into *OpenCV FaceDetector* sample code.

## Face Detection (cont.)

Features:

- Features are adapted from Haar basis functions.
- They are calculated by subtracting the sum of the pixels in the white from the sum of the pixels in the gray rectangles:

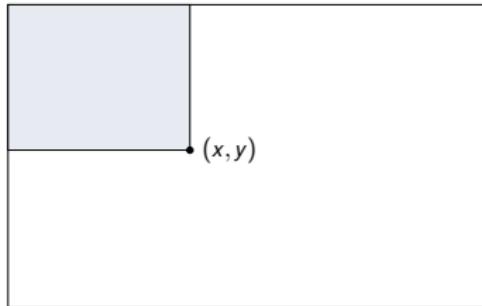
4 types of features used:



## Face Detection (cont.)

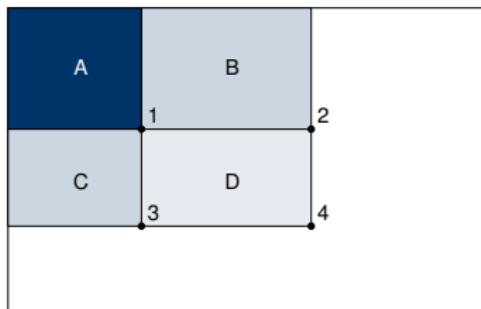
- The exhaustive set of features is very large: > 45 000 for a 380x280 image.
- Rectangle features can be efficiently computed using an Integral Image  $II$ :

$$II(x, y) = \sum_{x' \leq x, y' \leq y} I(x', y')$$



## Face Detection (cont.)

Computation of the sum of intensities for any rectangle D with just 3 basis operations based on the integral image:



- Value at 1: A
- Value at 2: A + B
- Value at 3: A + C
- Value at 4: A + B + C + D
- Sum within D:  $4 + 1 - (2 + 3)$

## Face Detection (cont.)

Boosting:

- The classification functions are restricted to each depend on a single feature only.
- This way, AdaBoost can be interpreted as an effective feature selecting algorithm:
  - The single rectangle feature is selected which best separates the observations.
  - For each feature, the weak learner determines the optimal threshold classification function.

## Face Detection (cont.)

- A weak classifier  $G_j(\mathbf{x})$  consists of a feature  $x_j$ , an optimal threshold  $\theta_j$ , and a parity  $s_j$  to indicate the direction of the inequality:

$$G_j(\mathbf{x}) = \begin{cases} 1 & \text{if } s_j f_j(\mathbf{x}) < s_j \theta_j \\ 0 & \text{otherwise} \end{cases}$$

with  $\mathbf{x}$  being a sub-window of an image.

## Face Detection (cont.)

- A weak classifier  $G_j(\mathbf{x})$  consists of a feature  $x_j$ , an optimal threshold  $\theta_j$ , and a parity  $s_j$  to indicate the direction of the inequality:

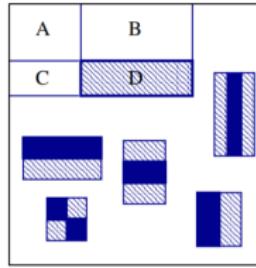
$$G_j(\mathbf{x}) = \begin{cases} 1 & \text{if } s_j f_j(\mathbf{x}) < s_j \theta_j \\ 0 & \text{otherwise} \end{cases}$$

with  $\mathbf{x}$  being a sub-window of an image.

- Very aggressive process to discard the vast majority of features.

## Face Detection (cont.)

- Best feature selected by AdaBoost:



Christian Hacker, Anton Batliner, and Elmar Noeth. Are You Looking at Me, Are You Talking with Me:  
Multimodal Classification of the Focus of Attention.

## Face Detection (cont.)

Classifier Cascade:

- Despite the efficiency of the feature computation, the complexity of the problem is still huge.
- Evaluating the full AdaBoost sequence on all sub-windows of the image takes too much time.

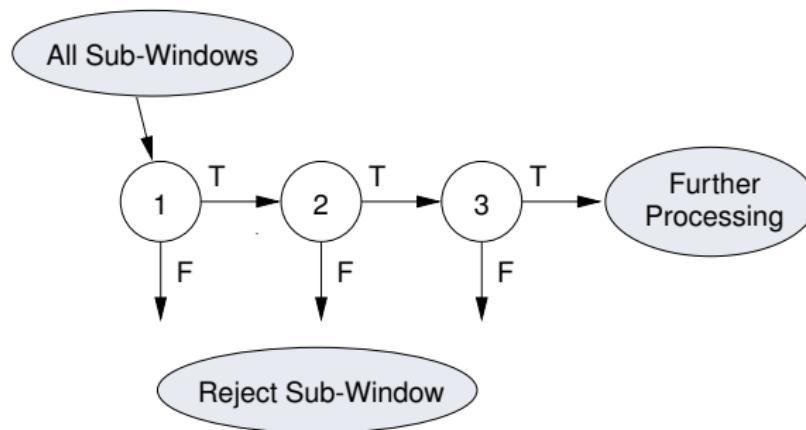
## Face Detection (cont.)

Classifier Cascade:

- Despite the efficiency of the feature computation, the complexity of the problem is still huge.
- Evaluating the full AdaBoost sequence on all sub-windows of the image takes too much time.
- Idea of cascaded classifiers:
  - Simpler classifiers used to reject the majority of sub-windows.
  - Each stage is again created by AdaBoost.
  - Stage 1 uses the two features shown before; detects 100 % faces with false positive rate (FPR) of around 40 %.

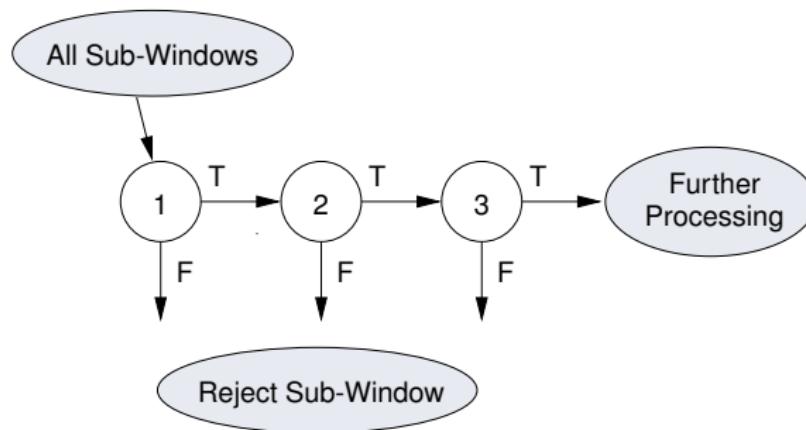
## Face Detection (cont.)

- The cascade has the overall shape of a degenerated decision tree:



## Face Detection (cont.)

- The cascade has the overall shape of a degenerated decision tree:



- A negative classification at any stage yields an immediate rejection of the sub-window.

## Face Detection (cont.)

- Structure of cascade reflects the (usually) overwhelming majority of negative sub-windows in an image.
- Goal is to reject as many negatives as possible at the earliest stage of the processing.

### Training of the cascade:

- Subsequent classifiers are trained using only those examples which pass through all the previous stages.
- The next classifier faces a more difficult task than the previous one.
- Trade-off between more features achieving higher detection rates and lower false positive rates while requiring more computation time.

## Face Detection (cont.)

Final classifier:

- 32 layer cascade of increasingly strong boosted classifiers

Layer	# Features	TPR	FPR
1	2	100 %	40 %
2	5	100 %	20 %
3-5	20		
6-7	50		
8-12	100		
13-32	200		

- Number of stages and features adapted until false positive rate on validation was nearly zero while maintaining high correct rate.

## Face Detection (cont.)

Example images from the training set:



Paul Viola and Michael Jones. Rapid Object Detection using a Boosted Cascade of Simple Features. IEEE Conf Comput Vis Pattern Recognit.

## Face Detection (cont.)

Results on test set:



Christian Hacker, Anton Batliner, and Elmar Noeth. Are You Looking at Me, Are You Talking with Me:  
Multimodal Classification of the Focus of Attention.

## Lessons Learned

- Boosting of weak classifiers can yield a powerful combined classifier
- AdaBoost algorithm
- Weights of the classifiers and on the data are adjusted
- AdaBoost minimizes an exponential loss function
- Viola & Jones face detection algorithm

## Further Reading

Most of the content of this lecture has been taken from:

- T. Hastie, R. Tibshirani, J. Friedman:  
*The Elements of Statistical Learning*,  
2nd Edition, Springer, 2009.
- Y. Freund, R. E. Schapire:  
*A decision-theoretic generalization of on-line learning and  
an application to boosting*,  
Journal of Computer and System Sciences, 55(1):119-139, 1997.
- P. A. Viola, M. J. Jones: *Robust Real-Time Face Detection*,  
International Journal of Computer Vision 57(2): 137-154, 2004.
- J. Matas and J. Šochman: *AdaBoost*,  
Centre for Machine Perception, Technical University, Prague.  
☞ [http://cmp.felk.cvut.cz/~sochmj1/adaboost\\_talk.pdf](http://cmp.felk.cvut.cz/~sochmj1/adaboost_talk.pdf)