

Ensemble Methods for Classifier Design

Bagging, Boosting, Arcing
Exponential Loss

Viola-Jones Face Detector &
Thunderstorm Prediction

Joachim M. Buhmann

November 21, 2019

Ensemble Learning

The idea of classifier ensembles

“Boosting is an approach to machine learning based on the idea of creating a highly accurate prediction rule by combining many relatively weak and inaccurate rules.”

R. Shapire, *Explaining AdaBoost*. In B. Schölkopf, Z. Luo, V. Vovk, eds, *Empirical Inference: Festschrift for Vladimir N. Vapnik*, Springer, 2013.

Computational advantage: Weak, but better than random classifiers are easy to train!

Statistical advantage: Data randomization in the spirit of Bootstrap captures statistical dependencies between alternative hypotheses.

⇒ The classifier ensemble encodes uncertainty intervals in the hypothesis class (output space).

Induction Principles for Classifier Selection

Rudolf Carnap: The problem of induction

Induction poses the central question:

“What justifies us in going from the direct observation of facts to a law that expresses certain regularities of nature?”

Empirical Risk Minimization (ERM) Principle

Select the classifier $c \in \mathcal{C}$ with the smallest error on the training data

$\mathcal{Z} = \{(x_1, y_1), \dots, (x_n, y_n)\}$:

$$\hat{c}_n^* \in \arg \min_{c \in \mathcal{C}} \hat{\mathcal{R}}_n(c) = \arg \min_{c \in \mathcal{C}} \frac{1}{n} \#\{c(x_i) \neq y_i : 1 \leq i \leq n\}$$

Bayesian inference by model averaging

Keep all classifiers according to their weights (posterior probabilities).

This strategy closely resembles ensemble methods.

Motivation for Ensemble Methods

Train several sufficiently diverse predictors, i.e., classifiers or regressors and use an aggregation scheme to produce a valid solution with low bias and variance.

Bagging or bootstrap aggregation combines several classifiers which have been trained on random subsamples (with replacements) of the data (Breiman, 1996)

Arcing or adaptive reweighting and combining is an extension of bagging introduced by Breiman (1998)

Boosting in form of AdaBoost is an ensemble method with error sensitive reweighting of the classifiers (Freund & Shapire, 1995).

Weak Learners Used for Bagging or Boosting

Stumps: Single axis parallel partition of space

Decision trees: Hierarchical partition of space

Multi-layer perceptrons: General non-linear function approximators

Radial basis functions: Non-linear expansions based on kernels

(In the following classifiers are mappings $c : \mathcal{O} \times \mathcal{X} \rightarrow \{-1, 1\}$.)

Combining Classifiers

Input: A pool of binary classifiers $c_1(x), c_2(x), \dots, c_B(x)$

Objective: Infer a composite classifier with weights $\{\alpha_b\}_{b=1}^B$

$$\hat{c}_B(x) = \text{sgn} \left(\sum_{b=1}^B \alpha_b c_b(x) \right)$$

Question: When can this procedure succeed?

Require **diversity** of the classifiers

Diversity: Use **different subsets** of the data for each c_b

Use **different features**

Decorrelate classifiers during training

Bagging Classifiers

Idea of bagging

Generate **diversity** in classifier pool by training on **different**, e.g. bootstrapped subsets!

Bootstrap sample: Given $\mathcal{Z} = \{(x_1, y_1), \dots, (x_n, y_n)\}$ generate \mathcal{Z}^* by choosing i.i.d. pairs **with replacement**.

INPUT: data $(x_1, y_1), \dots, (x_n, y_n)$

OUTPUT: classifier with **majority** vote of $\{c_1, c_2, \dots, c_B\}$

1: **for** $b = 1$ to B **do**

2: $\mathcal{Z}^{*b} =$ b -th bootstrap sample from \mathcal{Z}

3: Construct classifier c_b based on \mathcal{Z}^{*b}

4: **end for**

5: **return** ensemble classifier $\hat{c}_B(x) = \text{sgn} \left(\sum_{i=1}^B c_i(x) \right)$

Classifier selection: First compare, then bag!

Idea: compare classifiers $c'(x), c''(x)$ on bootstrap samples to find a winner.

Bootstrap sample: Given \mathcal{Z} generate \mathcal{Z}^* by sampling (x_i, y_i) with replacement.

INPUT: data $(x_1, y_1), \dots, (x_n, y_n)$

OUTPUT: classifiers $\hat{c}'_B(x), \hat{c}''_B(x)$ with **majority** votes of $\{c'_1, \dots, c'_B\}, \{c''_1, \dots, c''_B\}$

1: **for** $b = 1$ to B **do**

2: $\mathcal{Z}^{*b} = b^{\text{th}}$ bootstrap sample from \mathcal{Z}

3: Construct classifiers c'_b, c''_b based on \mathcal{Z}^{*b}

4: Calculate error difference $\Delta_b := \hat{\mathcal{R}}_n(c''_b) - \hat{\mathcal{R}}_n(c'_b)$

5: **end for**

6: **if** $(\text{median}(\Delta_1, \dots, \Delta_B) > 0)$ **then**

7: **return** ensemble classifier $\hat{c}'_B(x) = \text{sgn} \left(\sum_{i=1}^B c'_i(x) \right)$

8: **else**

9: **return** ensemble classifier $\hat{c}''_B(x) = \text{sgn} \left(\sum_{i=1}^B c''_i(x) \right)$

10: **end if**

Bagging: The Mechanism

Why does Bagging work? (A simple explanation)

Covariances small: Due to using **different** subsets for training

Variances similar: Estimator from each sub-sample behaves similarly (on average)

Biases: Weakly affected

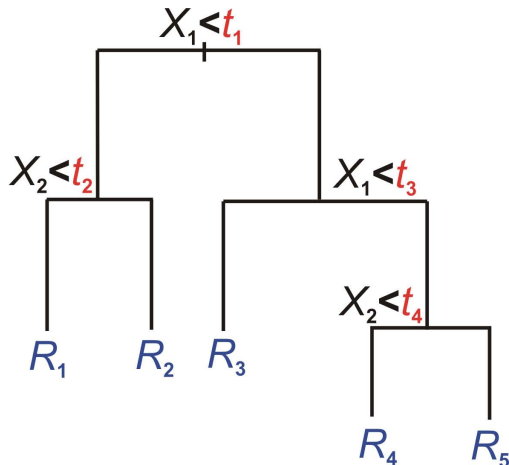
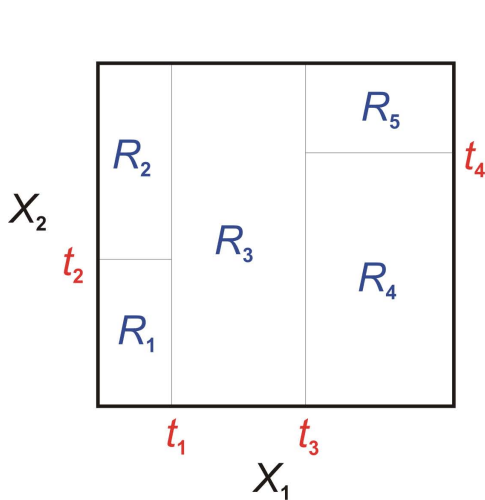
More elaborate explanation - Bühlmann and Yu 1999

Takeaway Messages:

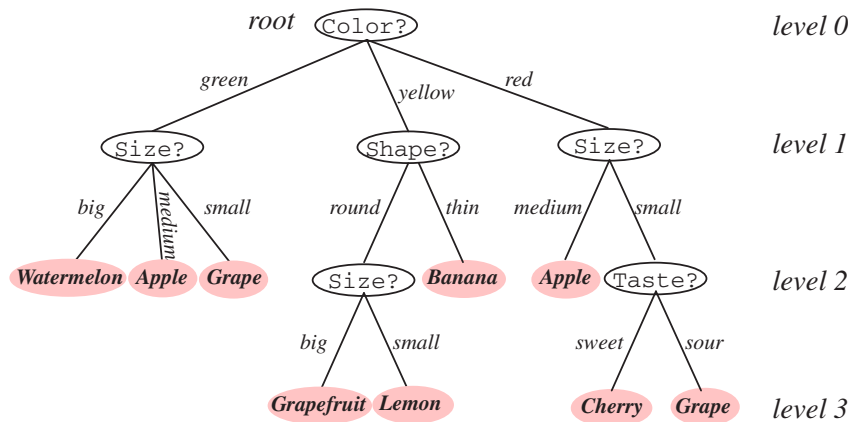
- ▶ It is advantageous to reduce dependence between component estimators
- ▶ Can we reduce bias **and** variance simultaneously?

Decision Trees

The data space is recursively partitioned by dichotomies. Widely used programs are CART or C4.5.



Decision Trees with Nominal Features



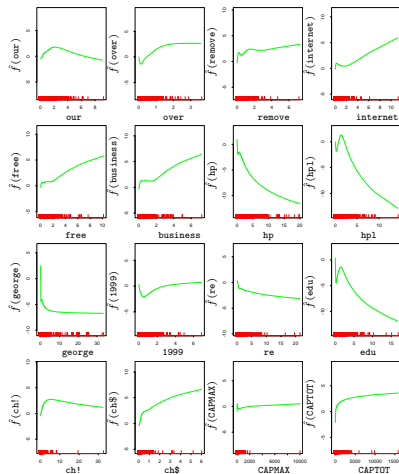
Example 1: Classification of fruits.

Richard O. Duda, Peter E. Hart, David G. Stork. *Pattern Classification*, 2nd Ed., Wiley, (2000)

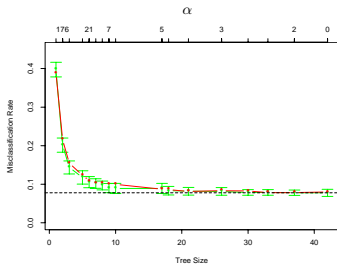
Example 2: Features for SPAM Filtering

Features:

- ▶ 48 quantitative predictors: percentage of word in an email that match word from a watch list.
- ▶ 6 quantitative predictors - percentage of $ch;$, $ch($, $ch[$, $ch!$, $ch\$$, $ch\#$
- ▶ CAPAVE, CAPMAX: average/maximal length of uninterrupted sequence of capitalized letters.
- ▶ CAPTOT: sum of the length of uninterrupted sequences of capitalized letters.



Example 3: Decision Trees for SPAM Filtering

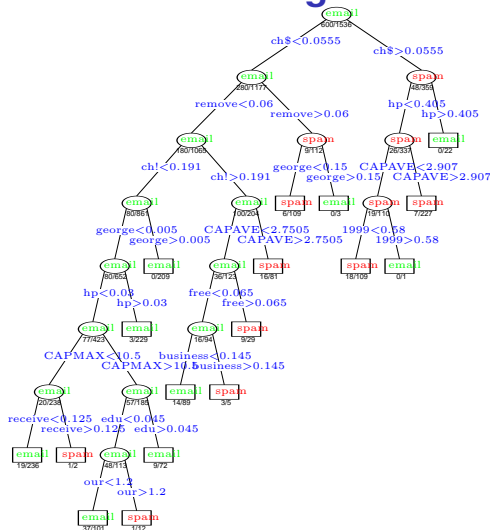


green curve

10 times crossvalidation

red curve

test error



Trevor Hastie, Robert Tibshirani, Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. 2nd Ed., Wiley, (2009)

Random Forests

Strategy for bagging trees

Grow a sufficiently deep decision tree to reduce bias \Rightarrow noisy classifier \Rightarrow average to enhance robustness

Classification of a new sample

Let $\hat{c}_b(x)$ be the class prediction of the b^{th} random forest tree. Then, the random forest classification yields

$$\hat{c}_{rf}^B(x) = \text{majority vote} \{ \hat{c}_b(x) \}_{b=1}^B$$

Example: biomarker detection with random forest classifiers.

(See additional slides on random forests for computational pathology.)

Random Forest algorithm

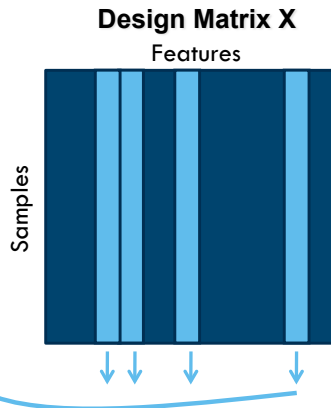
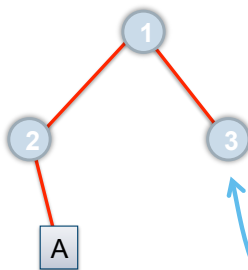
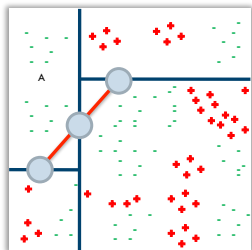
- 1: **for** $b = 1$ to B **do**
- 2: draw a bootstrap sample $\mathcal{Z}^* = \{(X_1^*, Y_1^*, \dots, (X_n^*, Y_n^*)\}$ of size n from the training data \mathcal{Z} ;

 // generation of a random forest tree:

 // grow a random forest tree $\hat{c}_b(x)$ to the bootstrapped data, by recursively repeating

 // the following steps for each leaf until the minimum node size n_{min} is reached.
- 3: **repeat**
- 4: select m variables at random from p variables ($m \approx \sqrt{p}$);
- 5: pick the best variable/split-point among the m selected variables;
- 6: split the node into two daughter nodes;
- 7: **until** node size n_{min} is reached
- 8: **end for**
- 9: **return** Output the ensemble of trees $\{\hat{c}_b(x)\}_{b=1}^B$

Random Forest Learning



At each node only a random subset of features is considered to choose the best split. Common splitting criteria are **entropy**, **Gini Index** and **misclassification rate**.

The Idea of Boosting

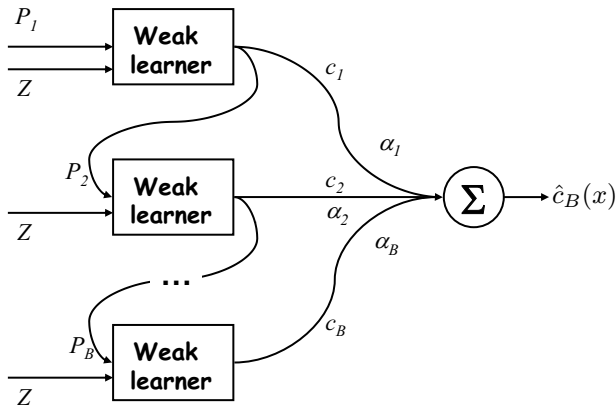
Basic idea: An **adaptive** combination of poor learners **with sufficient diversity** leads to an **excellent** (complex) classifier!

Base class: \mathcal{C} - **base class** of **simple** classifiers (e.g., perceptrons, decision stump, axis parallel splits)

Output Classifier: $\hat{c}_B(x) = \text{sgn} \left(\sum_{b=1}^B \alpha_b c_b(x) \right), c_b \in \mathcal{C}$

Idea outline: Train a sequence of simple classifiers on **modified** data distributions, and form a weighted average

AdaBoost



Weak Learner: $\epsilon_b \triangleq \sum_{i=1}^n w_i^{(b)} \mathbb{I}_{\{c_b(x_i) \neq y_i\}} / \sum_{i=1}^n w_i^{(b)}$ (c_b binary)

Data Reweighting

Data Modification

Apply weight $w_1^{(b)}, w_2^{(b)}, \dots, w_n^{(b)}$ to each of the training data (x_i, y_i) , $1 \leq i \leq n$ at the b -th boosting step.

$$\begin{aligned}\forall i \quad w_i^{(b+1)} &= w_i^{(b)} \exp(\alpha_b \mathbb{I}_{\{c_b(x_i) \neq y_i\}}) \\ w_i^{(1)} &= \frac{1}{n}\end{aligned}$$

$$\epsilon_b = \sum_{i=1}^n \frac{w_i^{(b)}}{\sum_{i=1}^n w_i^{(b)}} \mathbb{I}_{\{c_b(x_i) \neq y_i\}}$$

$$\alpha_b = \log \frac{1 - \epsilon_b}{\epsilon_b} > 0$$

AdaBoost Algorithm

AdaBoost.M1

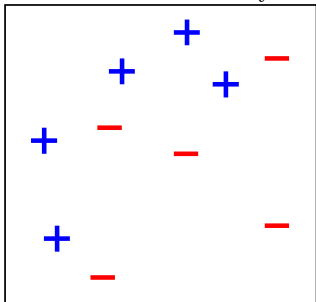
Input: data $(x_1, y_1), \dots, (x_n, y_n)$

Output: classifier $\hat{c}_B(x)$

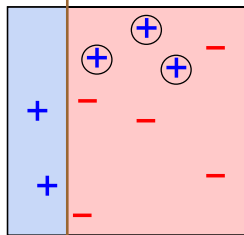
- 1: **initialize** observation weights $w_i = 1/n$
- 2: **for** $b = 1$ to B **do**
- 3: Fit a classifier $c_b(x)$ to the training data using weights w_i ;
- 4: Compute $\epsilon_b \leftarrow \sum_{i=1}^n w_i^{(b)} \mathbb{I}_{\{c_b(x_i) \neq y_i\}} / \sum_{i=1}^n w_i^{(b)}$;
- 5: Compute $\alpha_b \leftarrow \log \frac{1-\epsilon_b}{\epsilon_b}$;
- 6: Set $\forall i : w_i \leftarrow w_i \exp(\alpha_b \mathbb{I}_{\{y_i \neq c_b(x_i)\}})$;
- 7: **end for**
- 8: **return** $\hat{c}_B(x) = \text{sgn}\left(\sum_{b=1}^B \alpha_b c_b(x)\right)$

AdaBoost in Action I (Singer & Lewis Tutorial)

Data weights $w_i^{(1)}$



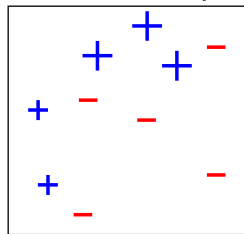
$c_1(x)$



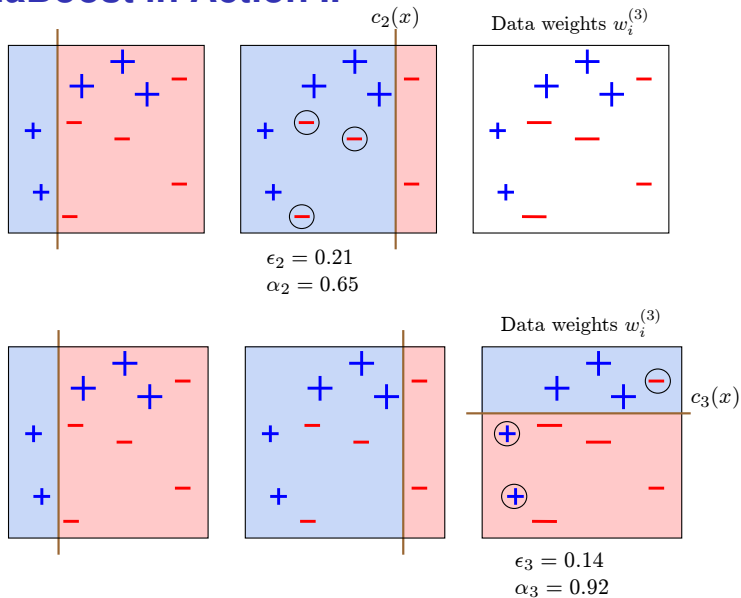
$$\epsilon_1 = 0.3$$

$$\alpha_1 = 0.42$$

Data weights $w_i^{(2)}$



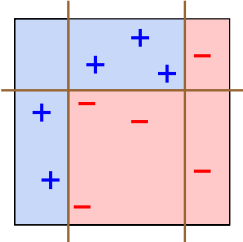
AdaBoost in Action II



Boosted Classifier

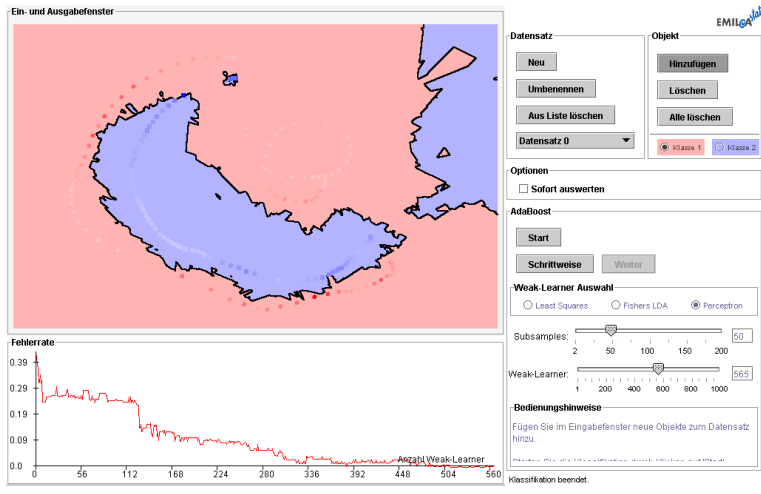
$$\hat{c}_{\text{final}} = \text{sgn} \left(0.42 \begin{array}{|c|c|} \hline \text{blue} & \text{red} \\ \hline \end{array} + 0.65 \begin{array}{|c|c|} \hline \text{blue} & \text{red} \\ \hline \end{array} + 0.92 \begin{array}{|c|c|} \hline \text{blue} & \text{red} \\ \hline \end{array} \right)$$

=



AdaBoost Applet

Applet HTML Page

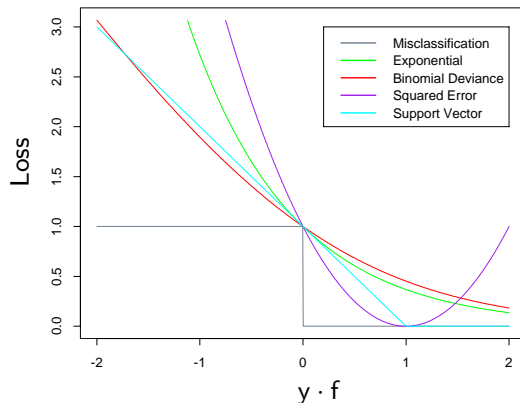


Theoretical Analysis of AdaBoost

J. Friedman, T. Hastie, R. Tibshirani, The Annals of Statistics 2000, Vol. 28, 337-407

- ▶ The **minimizer of the exponential loss function** (w.r.t. true distribution) is shown to be the **log-odds of the class probabilities**
- ▶ AdaBoost fits an additive model in base learners, optimizing the exponential loss function, i.e.,
AdaBoost.M1 is equivalent to forward stagewise additive modeling using exponential loss.

Loss functions for classification



Hastie, Tibshirani, Friedman (Figure 10.4): Loss functions for two-class classification. The response is $y = \pm 1$; the prediction is f , with class prediction $\text{sign}(f)$. The losses are misclassification: $\mathbb{I}\{\text{sign}(f) \neq y\}$; exponential: $\exp(-yf)$, binomial deviance: $\log(1 + \exp(-2yf))$, squared error: $(y - f)^2$, and support vector: $(1 - yf) \cdot \mathbb{I}\{yf > 1\}$. Each function has been scaled so that it passes through the point (0, 1).

Log-odds of class probabilities: minimizer of the exponential loss

Logistic regression: model the log posterior ratio of the two classes by an additive model $\sum_b c_b(x)$, i.e.,

$$\begin{aligned}\log \frac{\mathbf{P}\{y = 1|x\}}{\mathbf{P}\{y = -1|x\}} &= \sum_{b=1}^B c_b(x) =: F(x) \quad \Rightarrow \\ \mathbf{P}\{y = 1|x\} &= \frac{\exp(F(x))}{1 + \exp(F(x))}.\end{aligned}$$

AdaBoost is an additive logistic regression model! Consider minimizing the criterion $J(F)$ w.r.t. $F(x)$:

$$J(F) = \mathbb{E}\{\exp(-yF(x))\}$$

Lemma 1 $\mathbb{E}\{\exp(-yF(x))\}$ is minimized at

$$F(x) = \frac{1}{2} \log \frac{\mathbf{P}\{y = 1|x\}}{\mathbf{P}\{y = -1|x\}}.$$

Hence we derive the class posteriors

$$\mathbf{P}\{y = 1|x\} = \frac{\exp(F(x))}{\exp(-F(x)) + \exp(F(x))} = 1 - \mathbf{P}\{y = -1|x\}$$

Proof: \mathbb{E} is the expectation over (x, y) but it is sufficient to minimize the criterion conditioned on x .

$$\begin{aligned}\mathbb{E}\{e^{-yF(x)}|x\} &= \mathbf{P}\{y = 1|x\}e^{-F(x)} + \mathbf{P}\{y = -1|x\}e^{F(x)} \\ \frac{\partial \mathbb{E}\{e^{-yF(x)}|x\}}{\partial F(x)} &= -\mathbf{P}\{y = 1|x\}e^{-F(x)} + \mathbf{P}\{y = -1|x\}e^{F(x)} = 0 \\ \Rightarrow \mathbf{P}\{y = 1|x\} &= \frac{e^{2F(x)}}{1 + e^{2F(x)}} \quad \square\end{aligned}$$

AdaBoost as gradient descent

Theorem 2 The *Discrete AdaBoost* algorithm builds an additive logistic regression model via Newton-like updates for minimizing $\mathbb{E}\{e^{-yF(x)}\}$.

Proof: Let $J(F) = \mathbb{E}\{e^{-yF(x)}\}$. Suppose that $F(x)$ is the current estimate and $F(x) + \alpha c(x)$ is an improvement. For fixed α (and x), a second order Taylor expansion yields

$$\begin{aligned} J(F + \alpha c) &= \mathbb{E} \left\{ e^{-y(F(x) + \alpha c(x))} \right\} \\ &= \mathbb{E} \left\{ e^{-yF(x)} \left(1 - y\alpha c(x) + \frac{1}{2}\alpha^2 \underbrace{y^2 c(x)^2}_{=1} \right) \right\} + O(\alpha^3) \\ &= \mathbb{E} \left\{ e^{-yF(x)} \left(1 - y\alpha c(x) + \frac{1}{2}\alpha^2 \right) \right\} + O(\alpha^3) \end{aligned}$$

Pointwise minimization of exponential loss

Minimize $J(F + \alpha c)$ pointwise w.r.t. $c(x) \in \{-1, 1\}$

We introduce the abbreviation for the weighted expectation

$$\mathbb{E}_w\{g(x, y)|x\} := \frac{\mathbb{E}\{w(x, y)g(x, y)|x\}}{\mathbb{E}\{w(x, y)|x\}}, \quad w(x, y) = e^{-yF(x)}.$$

The condition for the optimal classifier yields (in $O(\alpha^2)$)

$$c(x) \in \arg \min_c \mathbb{E}_w\{1 - \alpha y c(x) + \alpha^2/2|x\} = \arg \max_c \mathbb{E}_w\{y c(x)|x\}.$$

Solution $c(x)$ is given by

$$c(x) = \begin{cases} 1 & \text{if } \mathbb{E}_w\{y|x\} = \mathbf{P}_w\{y = 1|x\} - \mathbf{P}_w\{y = -1|x\} > 0, \\ -1 & \text{otherwise.} \end{cases}$$

$$\mathbf{P}_w\{y = 1|x\} := \frac{\mathbf{P}\{y=1|x\} \exp(-F(x))}{\mathbf{P}\{y=1|x\} \exp(-F(x)) + \mathbf{P}\{y=-1|x\} \exp(F(x))} = \frac{\mathbf{P}\{y=1|x\}}{\mathbf{P}\{y=1|x\} + \mathbf{P}\{y=-1|x\} \exp(2F(x))} = 1 - \mathbf{P}_w\{y = -1|x\}$$

Newton-like update step

Interpretation of Newton-like update step

Consider the quadratic loss of classifier $c(x)$, i.e.,

$$\begin{aligned}\mathbb{E}_w\{(y - c(x))^2\}/2 - 1 &= \mathbb{E}_w\{(y^2 - 2yc(x) + (c(x))^2)\}/2 - 1 \\ &= -\mathbb{E}_w\{yc(x)\} \\ \Rightarrow \max_c \mathbb{E}_w\{yc(x)|x\} &= \min_c -\mathbb{E}_w\{yc(x)|x\} \\ &= \min_c \mathbb{E}_w\{(y - c(x))^2\}/2 - 1\end{aligned}$$

Thus minimizing a quadratic approximation to the criterion leads to a weighted least-squares choice of $c(x) \in \{-1, 1\} \Rightarrow$ **Newton-like step**.

Direct minimization of $J(F + \alpha c(x))$ w.r.t. α

$$\begin{aligned}\alpha^* &= \arg \min_{\alpha} \mathbb{E}_w \{ \exp(-\alpha y c(x)) \} \\ &= \arg \min_{\alpha} \left\{ e^{\alpha} \underbrace{\mathbb{E}_w \mathbb{I}_{\{y \neq c(x)\}}}_{\text{err}} + e^{-\alpha} \underbrace{\mathbb{E}_w \mathbb{I}_{\{y = c(x)\}}}_{1-\text{err}} \right\} = \frac{1}{2} \log \frac{1 - \text{err}}{\text{err}}\end{aligned}$$

Combination of these steps yields the **AdaBoost update**

$$\begin{aligned}F(x) &\leftarrow F(x) + \alpha^* c(x) = F(x) + \frac{1}{2} \log \left(\frac{1 - \text{err}}{\text{err}} \right) c(x) \\ w(x, y) &\leftarrow w(x, y) \exp(-\alpha^* y c(x)) \\ &= w(x, y) \exp \left(\log \left(\frac{1 - \text{err}}{\text{err}} \right) \mathbb{I}_{\{y \neq c(x)\}} \right) \exp(-\alpha^*)\end{aligned}$$

Note that the identity $-yc(x) = 2\mathbb{I}_{\{y \neq c(x)\}} - 1$ holds. The resulting constant factor $\exp(-\alpha^*)$ can be neglected since it does not change the effect of the weights.

Robust Real-Time Face Detection

P. Viola, M. Jones, Int. J. Computer Vision 57(2), 137-154, (2004)

Goal

Build a real-time (15 frames per second) face detection system with competitive detection rates.

System components

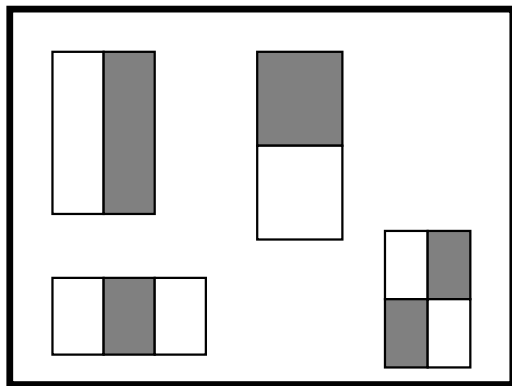
1. image representation called “Integral Images”;
2. efficient classifier trained by Adaboost to select a small number of features from a large set of potential features;
3. a method for combining classifiers in a **cascade** which quickly discards background regions of the image while spending more computation on face-like regions.

Features for Face Detector

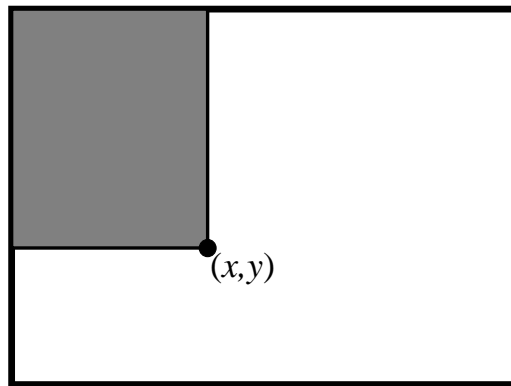
Feature design

Two-rectangle features as the difference between the sum of the pixels within two rectangular regions.

rectangle features



integral image features



Feature Selection by AdaBoost

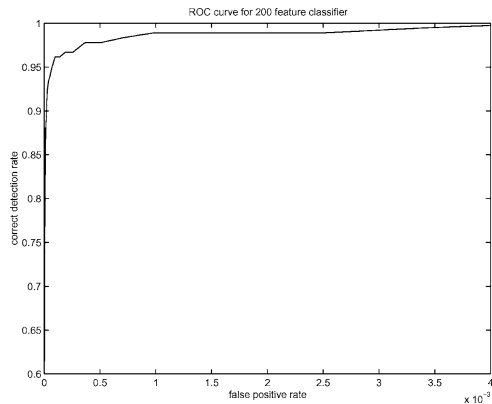
Learning scenario: select the features and train the classifier simultaneously.

Weak learner: decision stump for the 24×24 sub-window x of an image with feature $f(x)$ and polarity $p \in \{-1, 1\}$

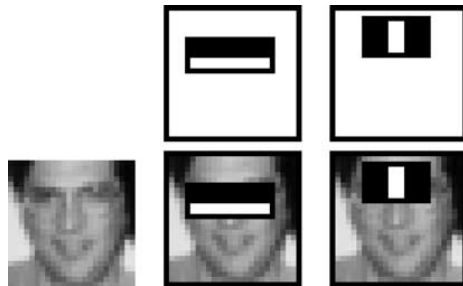
$$c(x, f, p, \theta) = \begin{cases} 1 & \text{if } pf(x) \leq p\theta \\ 0 & \text{otherwise} \end{cases}$$

early selected features with error rates ≈ 0.1 , late features with rates in $[0.4, 0.5]$.

Learning results: 200 features yielded 95% detection rate at 1/14084 false positive rate (target rate 10^{-6} for real world applications).

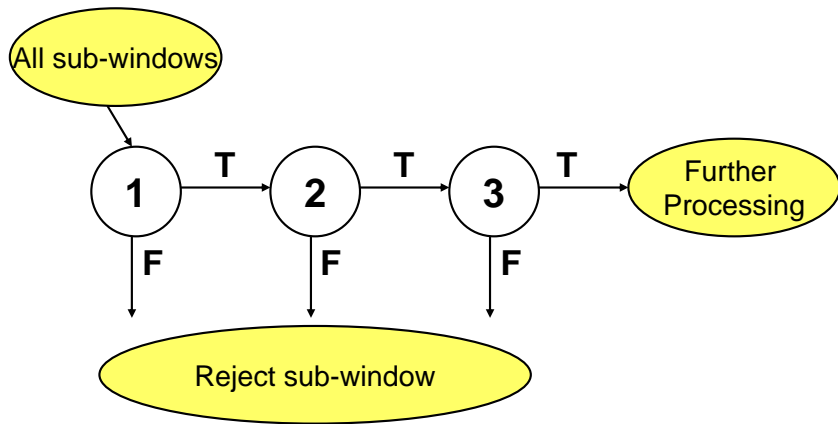


Receiver operating characteristic (ROC) curve for the 200 feature classifier.

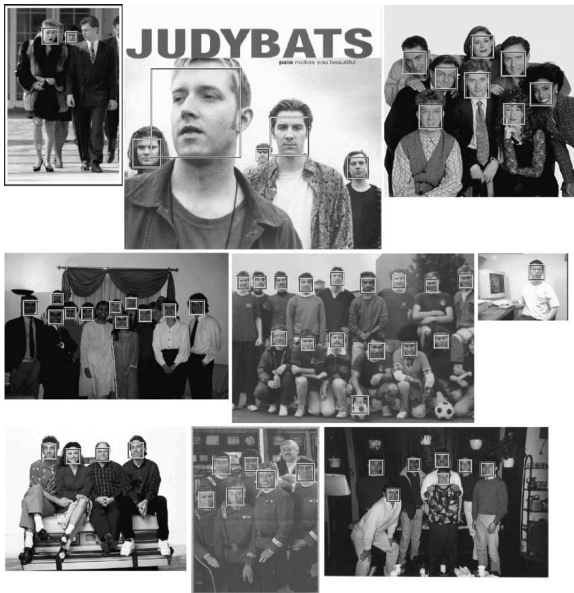


The first and second features selected by Adaboost. The first feature measures the difference in intensity between the regions of the eyes and the upper cheeks. The second feature compares the intensities in the eye regions and at the bridge of the nose.

Attentional Cascade



Schematic depiction of the detection cascade. A series of classifiers, applied to every sub-window, eliminate a larger and larger number of negative examples with increasing processing load.



Boosting for Thunderstorm Prediction

(Donat Perler, MS thesis, D-INFK, ETHZ, 2006)

MeteoSwiss weather forecast uses Adaboost in a pilot study for thunderstorm prediction.

Expert system of German weather forecast DWD is clearly outperformed.



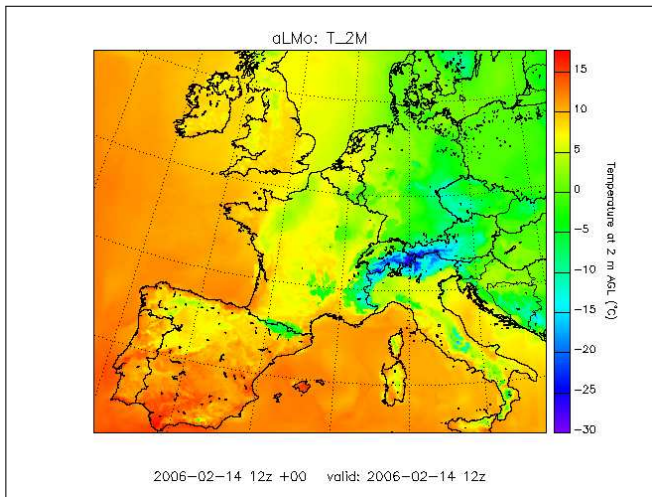
Data for Weather Forecast

Analysis Data from the Swiss numerical weather model `alpine model` provides input for classifiers: The analysis data are the output of the weather model at lead time zero, they are available every hour covering central Europa.

Present Weather fields from SYNOP-Messages: These fields are weather codes based on human observations from hundreds of stations over Europe 5-8 times a day.

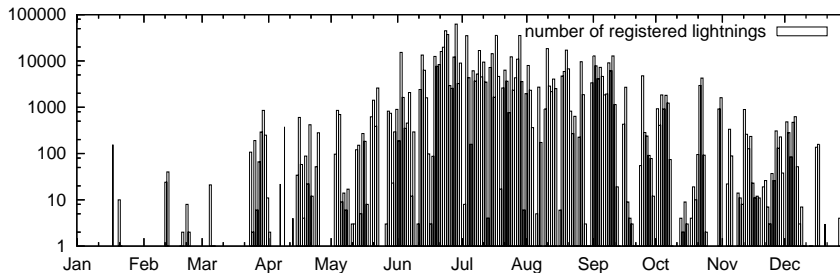
Lightning Data from the lightning detection system EUCLID: data is obtained from MeteOrage, a member of the EUCLID Network. The system detects lightning over Europe.

Temperature at 2 m



MeteoSwiss Analysis of the 2 m temperature over the whole aLMO domain for 2006-02-14 12:00 UTC.

Lightening Distribution in 2005



Day in the year 2005 versus the number of registered lightnings in the MeteoSwiss data-warehouse.

Features for Thunderstorm Forecast

Aversion	Description
LAT	Latitude
LONG	Longitude
HEIGHT	Height above sea level
DATE	Day of year
TIME	Day time
PMSL	Mean sea level pressure
T★	Temperature on the {500, 700, 850, 950} hPa model reference level
WDIR★	Horizontal wind direction on the {500, 700, 850, 950} hPa model reference level
WV★	Horizontal wind velocity on the {500, 700, 850} hPa model reference level
VERTW★	Vertical wind velocity on the {500, 700, 850, 950} hPa model reference level
RELHUM★	Relative humidity on the {500, 700, 850} hPa model reference level
THETA★	Equivalent potential temperature on the {500, 700, 850, 950} hPa model reference level
TD850	Dew point temperature on the 850 hPa model reference level
WSHEARL	Vertical wind shear between surface and 3 km above surface
WSHEARM	Vertical wind shear between surface and 6 km above surface

Aversion	Description
WSHEARU	Vertical wind shear between 3 km and 6 km above surface
WDIRVAR	Variance of the wind direction between 950 hPa and 500 hPa model reference level
RG	Total precipitation
RK	Convective precipitation
PDIFF	Pressure difference between cloud base and top
TTOP	Temperature at cloud top
KOI	KO-index
TT	TT-index (Total Totals Index)
SWEAT	SWEAT-index (Severe WEather Thread index)
SHOWI	Showalter-index
LI	Surface lifted index
DCI	DCI-index (Deep Convection Index)
CAPE	Convective available potential energy
ADEDO2	Adedokun ₂
MOCON	Surface moisture flux convergence
MOCONI	Surface moisture flux convergence integrated over the lowest 100 hPa
ADTHE	Equivalent potential temperature advection
HSURF	Model height of the surface over sea level

Forecast Results

Comparison with expert system of the German Weather Service (DWD)

Compare DWD's expert system with boosted decision stump classifiers operating on the same feature space

Classifier	POD	FAR	FBI	CSI	HSS
DWD's expert system	18%	94%	3.12	0.05	0.08
Decision stumps	45%	68%	1.42	0.23	0.34
AdaBoost	57%	59%	1.44	0.32	0.46

POD: probability of detection, FAR: false alarm ratio, FBI: frequency bias, CSI: critical success index, HSS: Heidke skill score

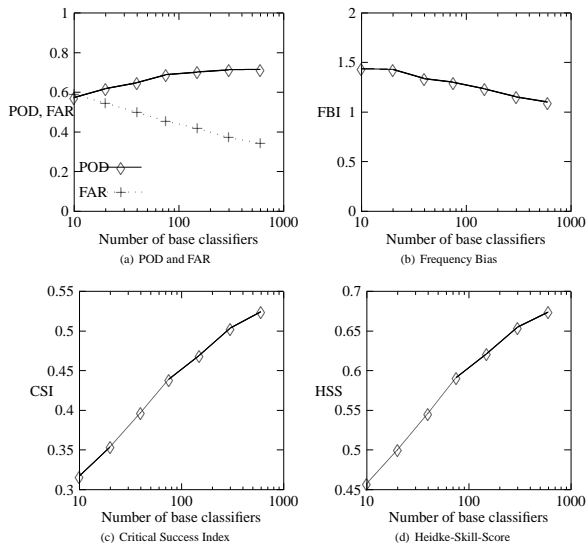
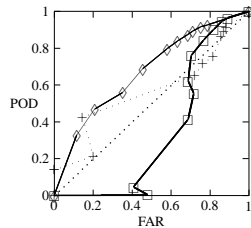
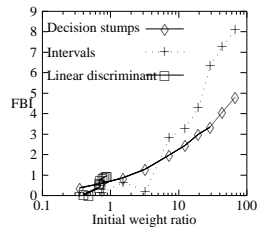


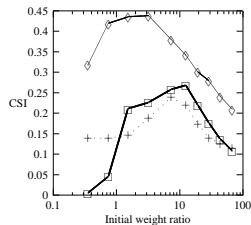
Figure 6.2: The number of base classifiers influences the skill scores.



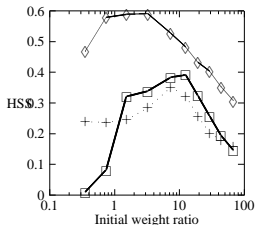
(a) ROC-curve



(b) Frequency Bias



(c) Critical Success Index



(d) Heidke-Skill-Score

AdaBoost scores with different base classifiers and initial weight ratios.

Comparison of Bagging and Boosting

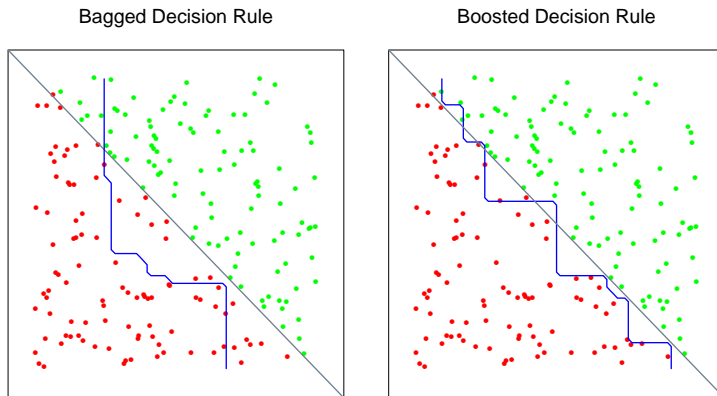
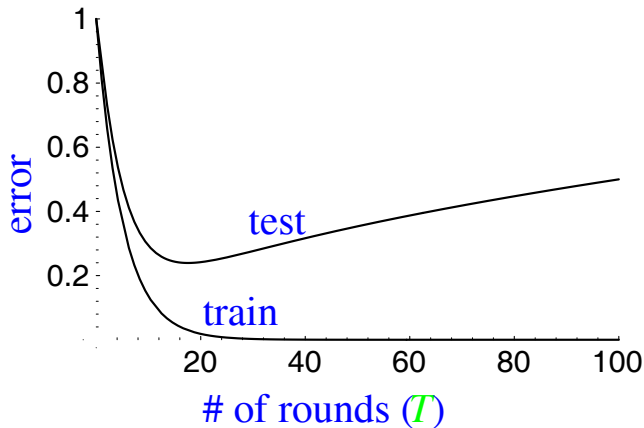


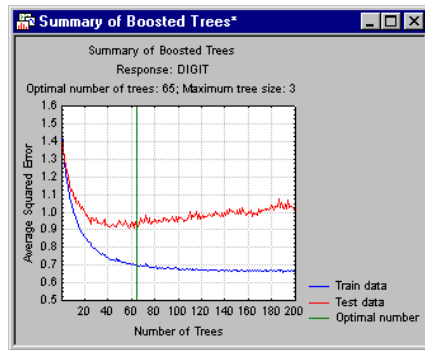
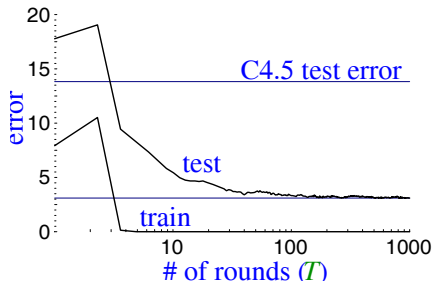
Figure 8.11 (HTF): Left: decision boundary estimated from bagging the decision rule from a single split, axis-oriented classifier (test error rate 0.166). Right: decision boundary from boosting the decision rule of the same classifier (test error rate 0.065).

Theoretical Learning Curves



Usually, the test error initially decreases for iterative adaptation procedures and then it increases again when overfitting. The training error, however, will not increase.

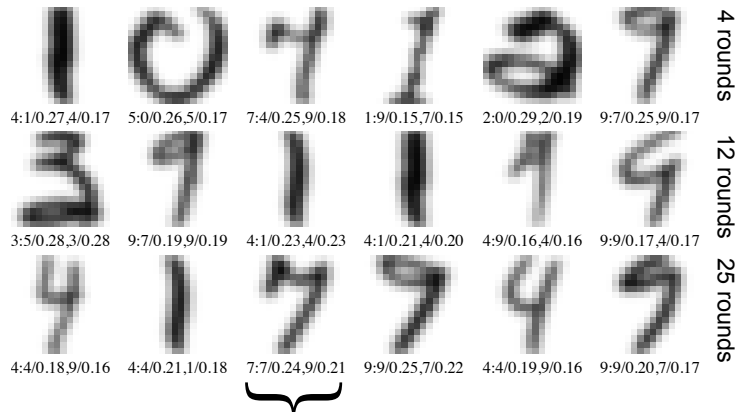
Experiments with C4.5 Decision Trees



The experiment on the left shows the astonishing behavior that the test error does not increase after reaching a minimal value. Researchers initially suspected that boosting prevents overfitting in general. Later, it has been demonstrated that overfitting can occur as seen on the right (source: <http://www.statsoft.com/textbook/stbootres.html>). However, the boosting algorithm shows a remarkable robustness and overfitting effects are often “relatively benign”.

Digit recognition with AdaBoost

Examples that have the largest weights at round 4, 16, 25



Label of sample : 1st label / score, 2nd label / score

Yoav Freund, Robert E. Schapire. Experiments with a new boosting algorithm. ICML 1996, 148–156.