An Introduction to Boosting
Part I

Gunnar Rätsch

Friedrich Miescher Laboratory
of the Max Planck Society

http://www.fml.tuebingen.mpg.de/~raetsch

Help with earlier presentations: Ron Meir, Klaus-R. Müller

Contents of this Course

- Part I: Introduction to Boosting
  1. Basic Issues, PAC Learning
  2. The Idea of Boosting
  3. AdaBoost's Behavior on the Sample
- Part II: Some Theory
  4. Generalization Error Bounds
  5. Boosting and Large Margins
  6. Leveraging as Greedy Optimization
- Part III: Extensions and Applications
  7. Boosting vs SVMs
  8. Improving Robustness & Extensions
  9. Applications

Slides available from the wiki and from

`http://www.fml.mpg.de/~raetsch/boostintro`

Sources of Information

Internet http://www.boosting.org*
http://www.cs.princeton.edu/~schapire/boost.html

Conferences Computational Learning Theory (COLT), Neural Information Processing Systems (NIPS), Int. Conference on Machine Learning (ICML), ...

Journals Machine Learning, Journal of Machine Learning Research, Information and Computation, Annals of Statistics

People List available at http://www.boosting.org

Software Only few implementations: algorithms are "too simple" (cf. http://www.boosting.org)

Review Papers Schapire, 1999, and Meir & Rätsch, 2002.

* New version: http://www3.fml.tuebingen.mpg.de/boosting

Learning – Problem Formulation I



The 'World':

Data: $\{(\mathbf{x}_n, y_n)\}_{n=1}^N, \mathbf{x}_n \in \mathbf{R}^d, y_n \in \{\pm 1\}$

Unknown target function: $y = f(\mathbf{x})$ (or $y \sim P(y|\mathbf{x})$)

Unknown distribution: $\mathbf{x} \sim p(\mathbf{x})$

Objective: Given new $\mathbf{x}$, predict y

Problem: $P(\mathbf{x}, y)$ is unknown!

Learning – Problem Formulation II

The 'Model'

Hypothesis class: $\mathcal{H} = \{h \mid h : \mathbf{R}^d \to \{\pm 1\}\}$

Loss: $l(y, h(\mathbf{x}))$ (e.g. $\mathbf{I}[y \neq h(\mathbf{x})]$)

Objective: Minimize the true (expected) loss – "generalization error"

$$h^* = \operatorname*{argmin}_{h \in \mathcal{H}} L(h) \text{ with } \boxed{L(h) := \mathbf{E}_{\mathbb{X} \times \mathbb{Y}}\, l(\mathbb{Y}, h(\mathbb{X}))}$$

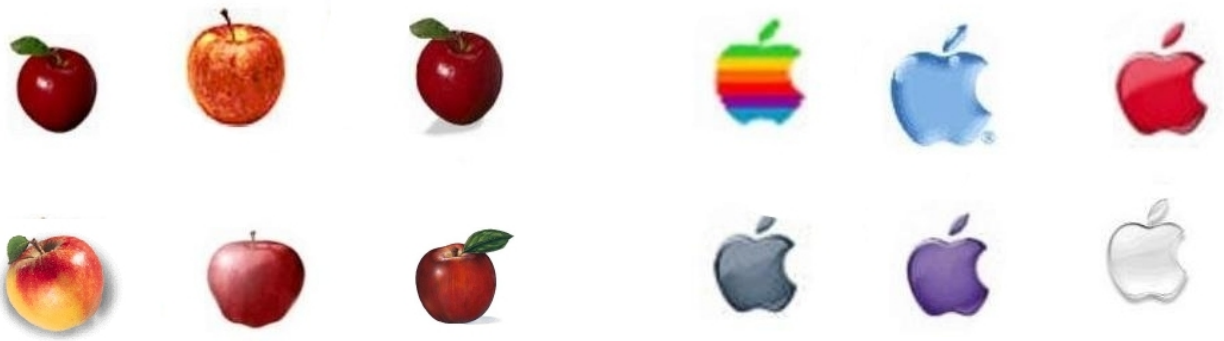
Problem: Only have a data sample available, $P(\mathbf{x}, y)$ is unknown!

Solution: Find empirical minimizer

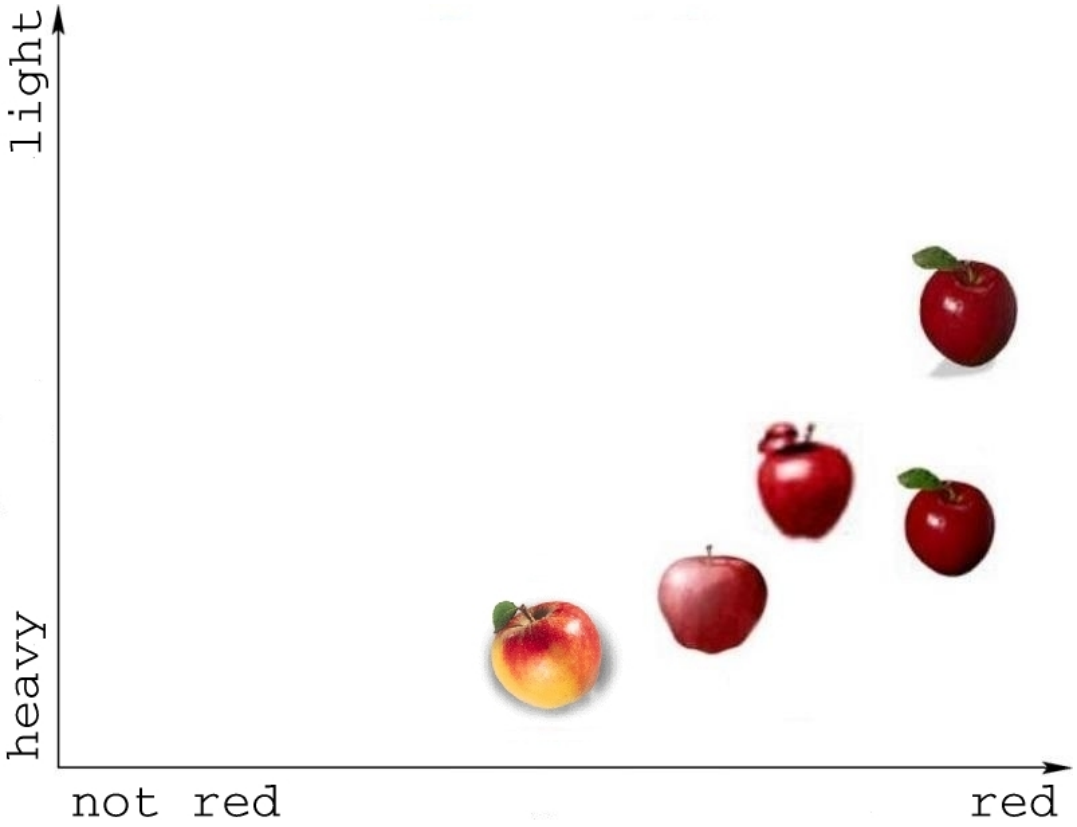
$$\hat{h}_N = \operatorname*{argmin}_{h \in \mathcal{H}} \frac{1}{N} \sum_{n=1}^{N} l(y_n, h(\mathbf{x}_n))$$

How can we efficiently construct complex hypotheses with small generalization errors?

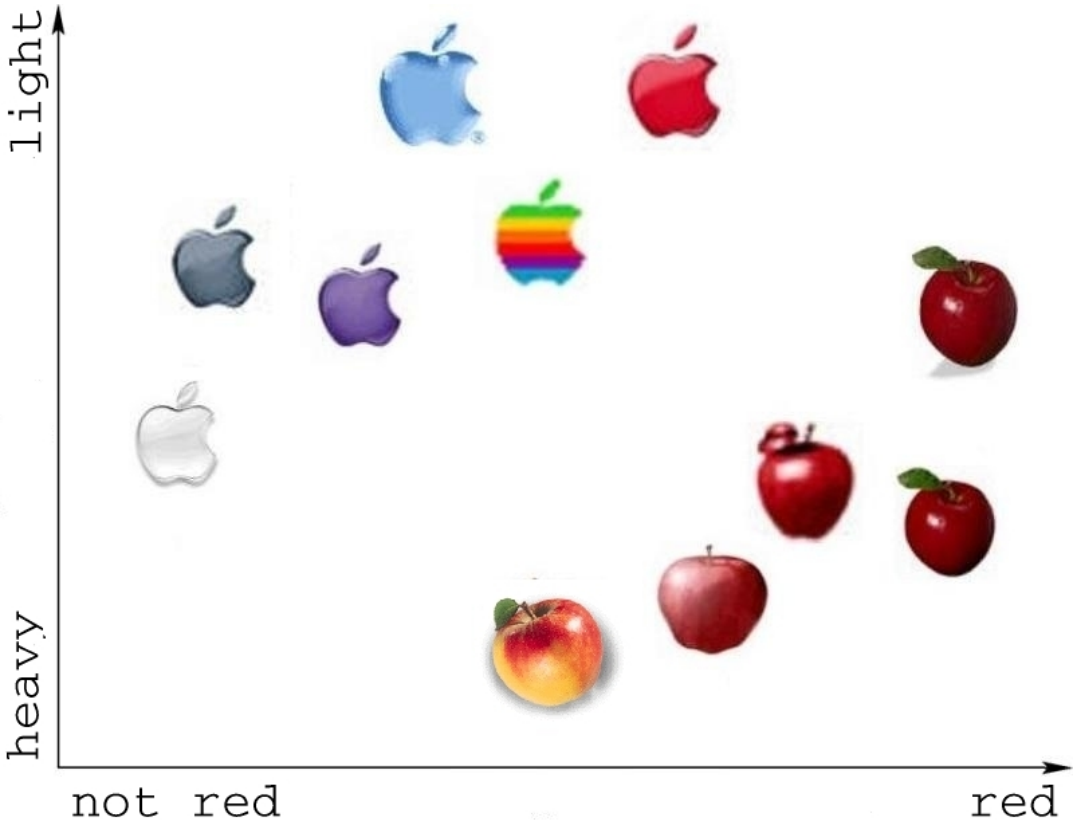
Example: natural vs. plastic Apples



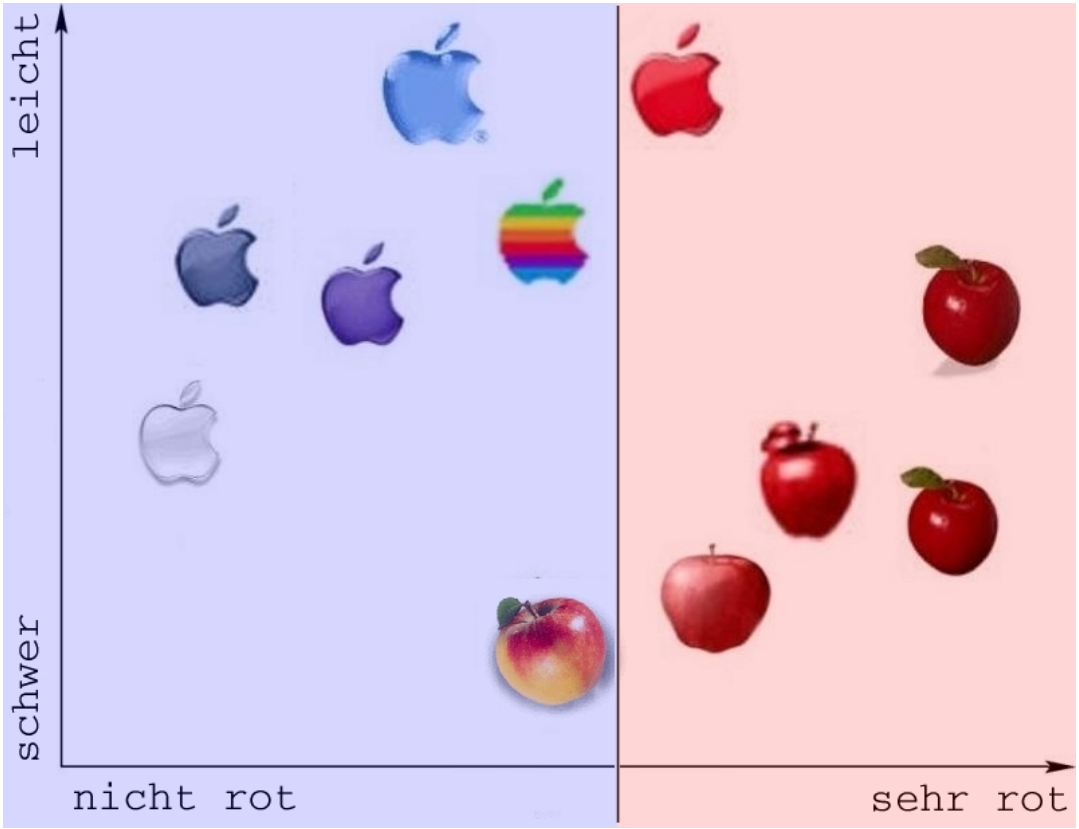
Example: natural vs. plastic Apples



Example: natural vs. plastic Apples



Simple Hypotheses: Cuts on Coordinate Axes



PAC Learning

Input: Sample $S_N = (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_n) \sim P(\mathbf{x}, y)$
Hypothesis class $\mathcal{H} \ni h : \mathbf{x} \mapsto y$
Accuracy parameter ϵ
Confidence parameter δ

Algorithm a mapping from S_N to $\mathcal{H}$, $\mathcal{L} : S_N \mapsto \hat{h}_N$

Output function $\hat{h}_N$ from $\mathcal{H}$

Requirements 1. **P**robably **A**pproximately **C**orrect:

$$\mathbf{Pr}\left(S_N : L(\hat{h}_N) - L(h^*) \geq \epsilon\right) \leq \delta$$

where $h^* = \operatorname{argmin}_{h \in \mathcal{H}} \mathbf{E}\, l(\mathbb{Y}, h(\mathbb{X}))$

2. "efficient":
$\Rightarrow$ polynomial runtime in $N, 1/\epsilon, 1/\delta$

Levels of Generality

Recall

$$h^* = \operatorname*{argmin}_{h \in \mathcal{H}} L(h)$$
$$h^B = \operatorname*{argmin}_{h} L(h) \qquad \text{(h unrestricted)}$$

Restricted Setting: Assume $h^B = h^* \in \mathcal{H}$, require

$$L(\hat{h}_N) \xrightarrow{P} L(h^*)$$

Agnostic Setting: Assume nothing about h^B, require

$$L(\hat{h}_N) \xrightarrow{P} L(h^*)$$

Universal Setting: Assume nothing about h^B, require

$$L(\hat{h}_N) \xrightarrow{P} L(h^B)$$

Weak and Strong Learning

Assumption: Restricted setting, i.e. $h^B \in \mathcal{H}$

Strong PAC Learning: Demand small error with high probability:

$$\mathbf{Pr}\left(S_N : L(\hat{h}_N) - L(h^*) \geq \epsilon\right) \leq \delta \qquad (1)$$

for arbitrary (small) ϵ and δ

Weak PAC Learning: Demand that (1) holds for ϵ and δ

- Example: Binary classification, require that error ϵ is smaller than $1/2$:

$$\epsilon \leq \frac{1}{2}(1-\gamma) \qquad \text{for some fixed } \gamma > 0$$

With Boosting one can obtain a strong learner from a weak learner!

2. The Idea of Boosting

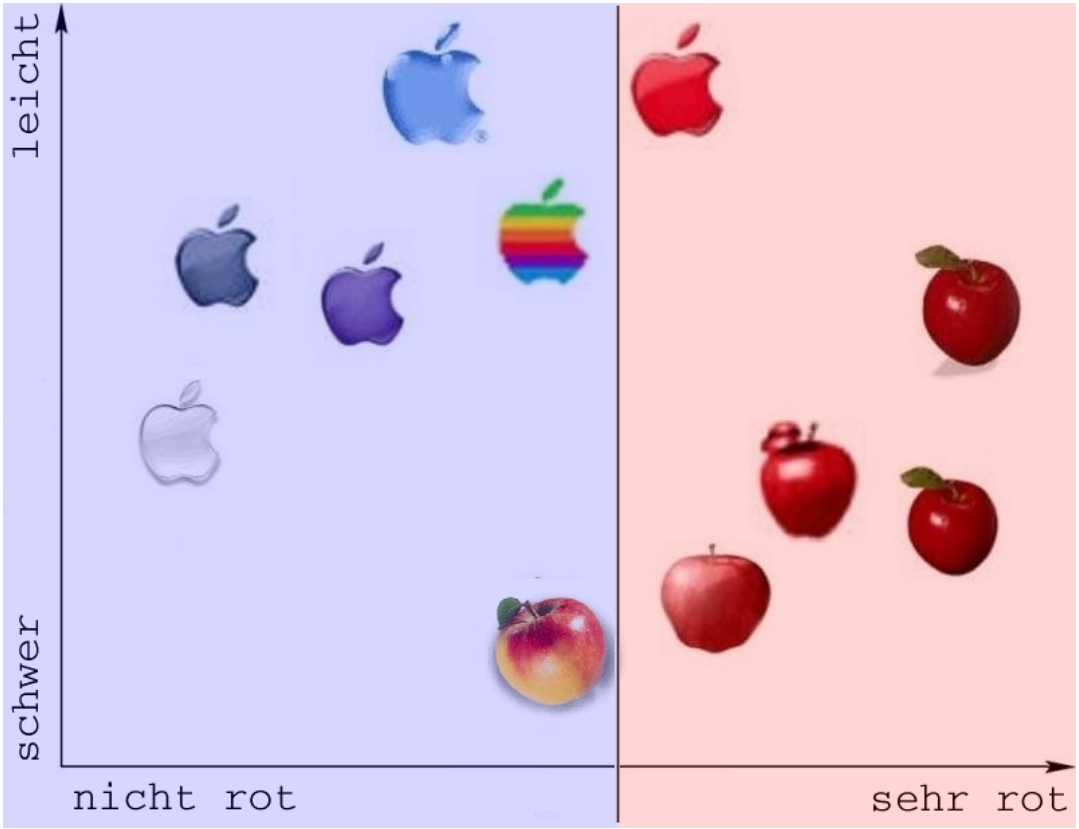


Hill Inlet, Whitsunday Island

Main Sources:

- Freund 1995, Freund & Schapire 1996
- Schapire, Freund, Bartlett & Lee 1998
- Breiman 1996

Simple Hypotheses: Cuts on Coordinate Axes

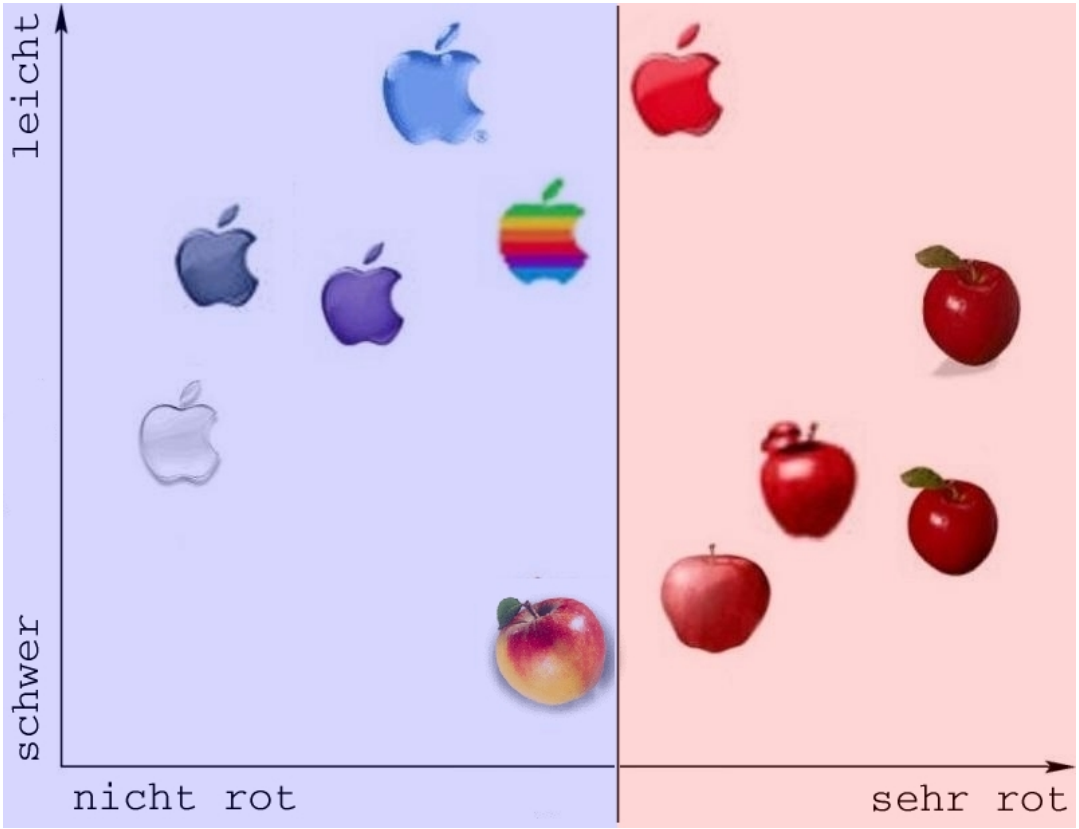


AdaBoost (Freund & Schapire 1996)

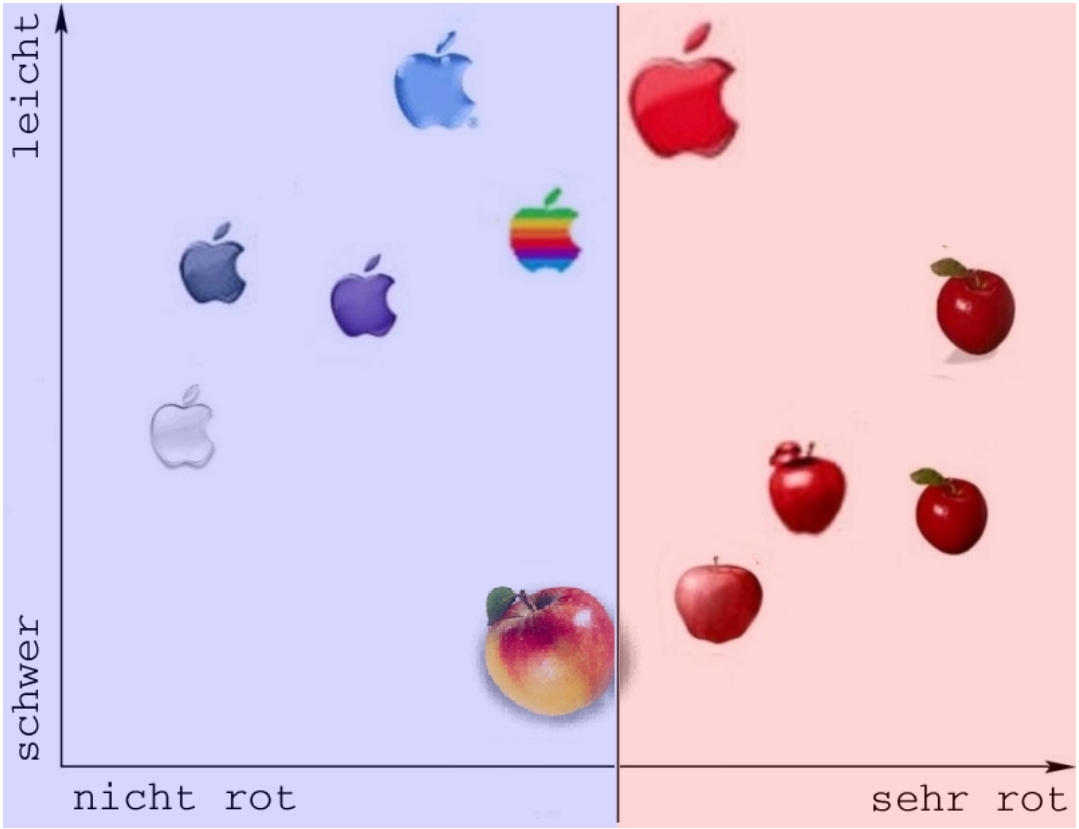
- **Idea:**
  - Simple hypotheses are not perfect!
  - Hypotheses combination $\Rightarrow$ increased accuracy
- **Problems:**
  - How to generate different hypotheses?
  - How to combine them?
- **Method:**
  - Compute distribution $d_1, \ldots, d_N$ on examples
  - Find hypothesis on the weighted training sample $(\mathbf{x}_1, y_1, d_1), \ldots, (\mathbf{x}_N, y_N, d_N)$
  - Combine hypotheses $h_1, h_2, \ldots$ linearly:

$$f = \sum_{t=1}^{T} \alpha_t h_t$$

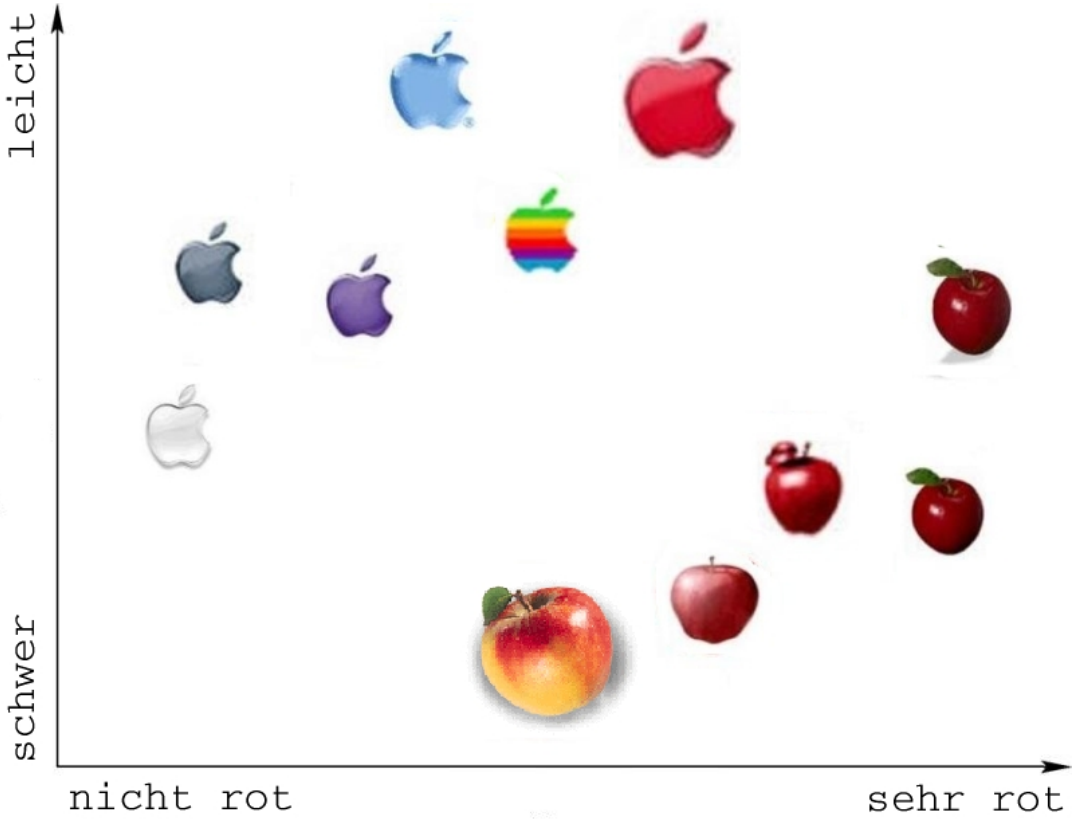
Boosting: 1st Iteration



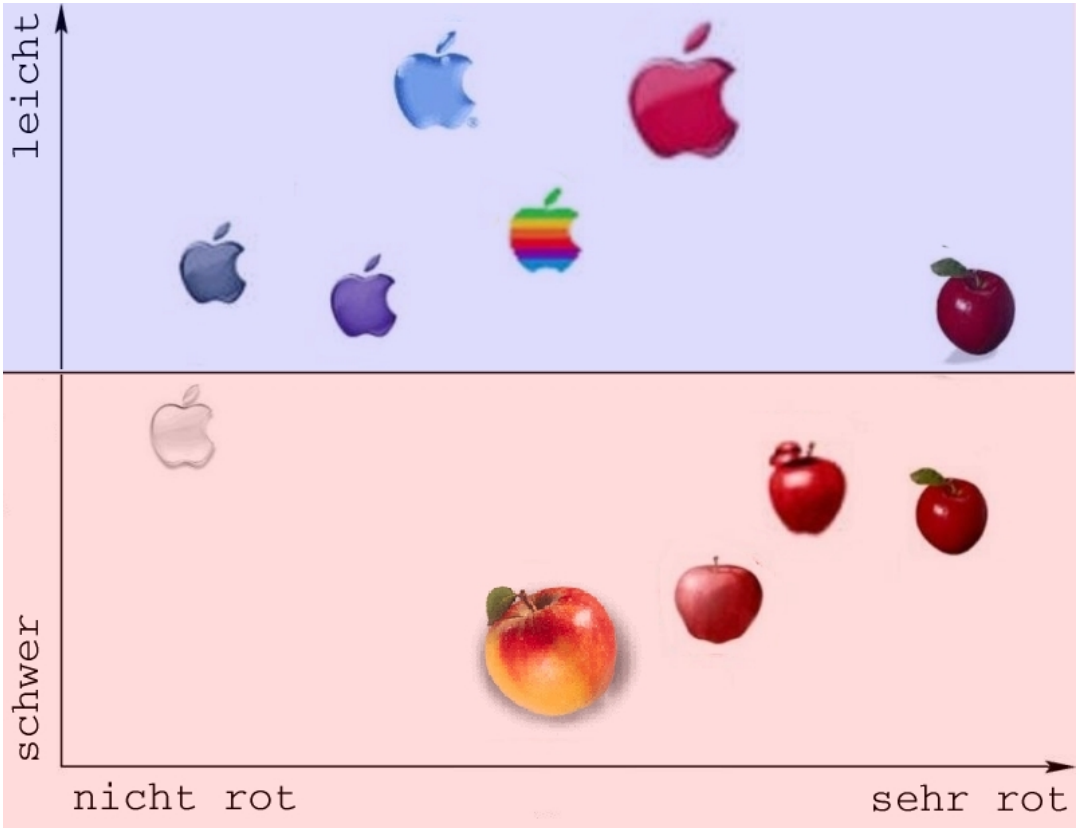
Boosting: Recompute weighting



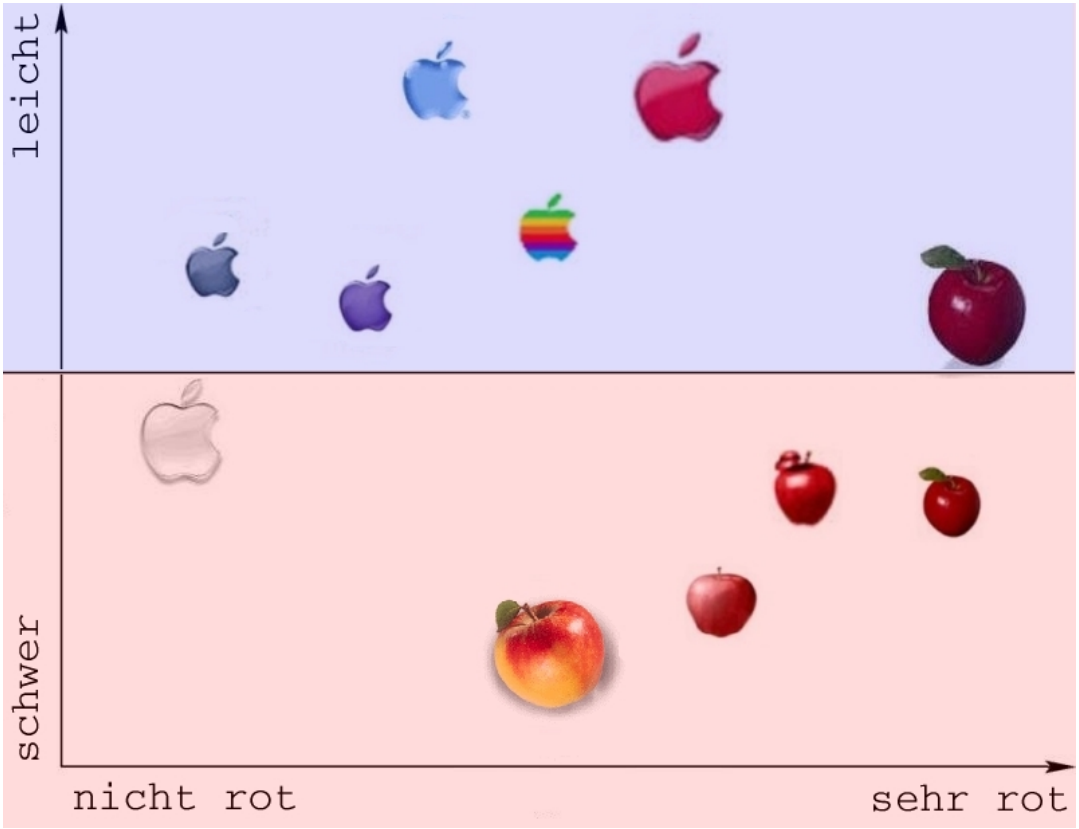
Boosting: 2nd Iteration



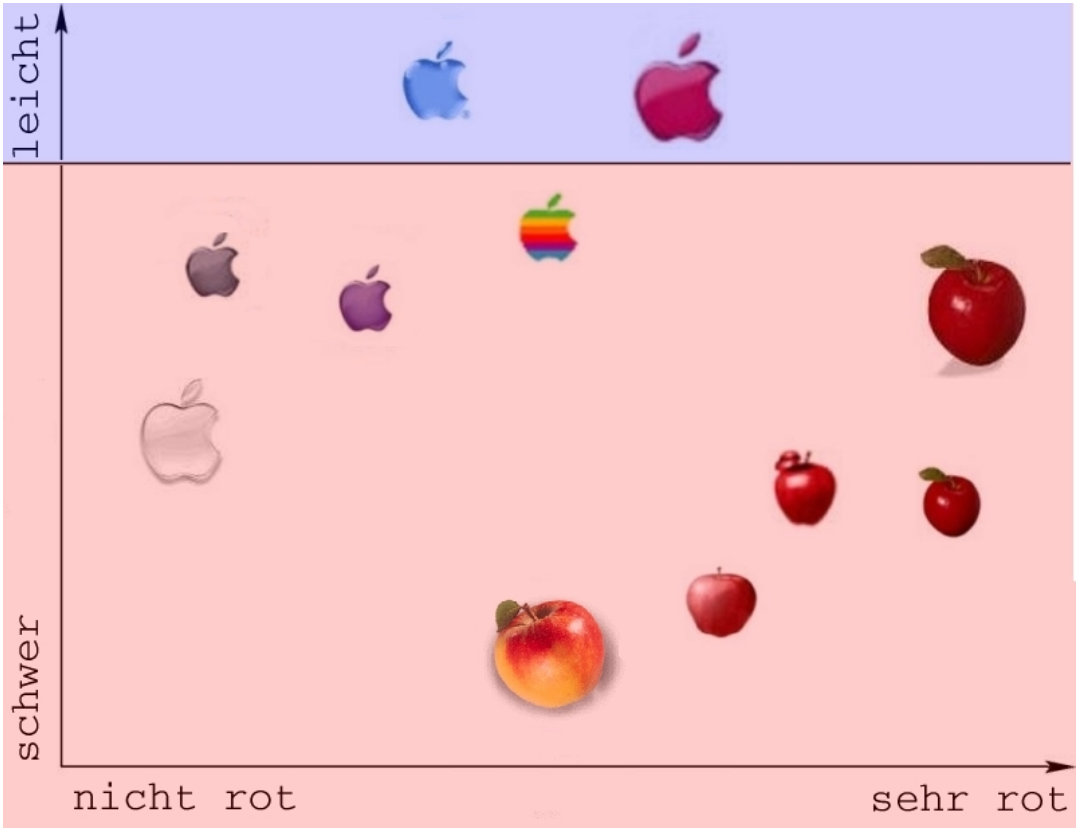
Boosting: 2nd Hypothesis



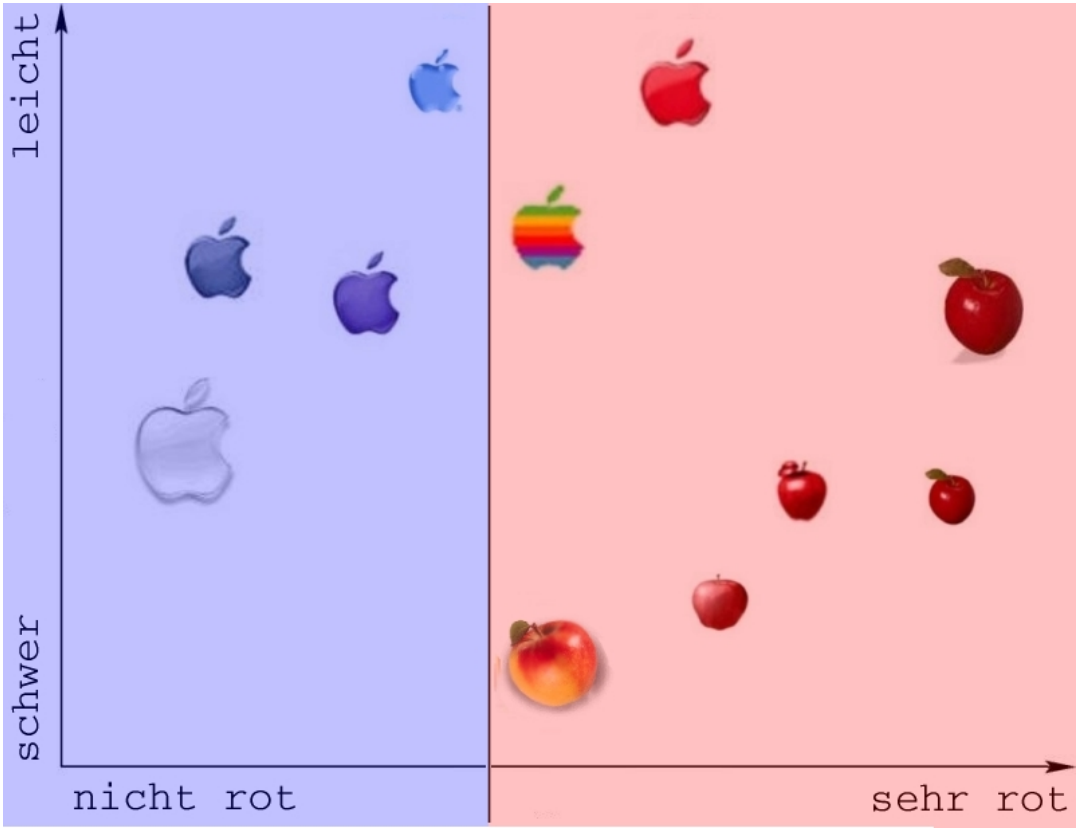
Boosting: Recompute weighting



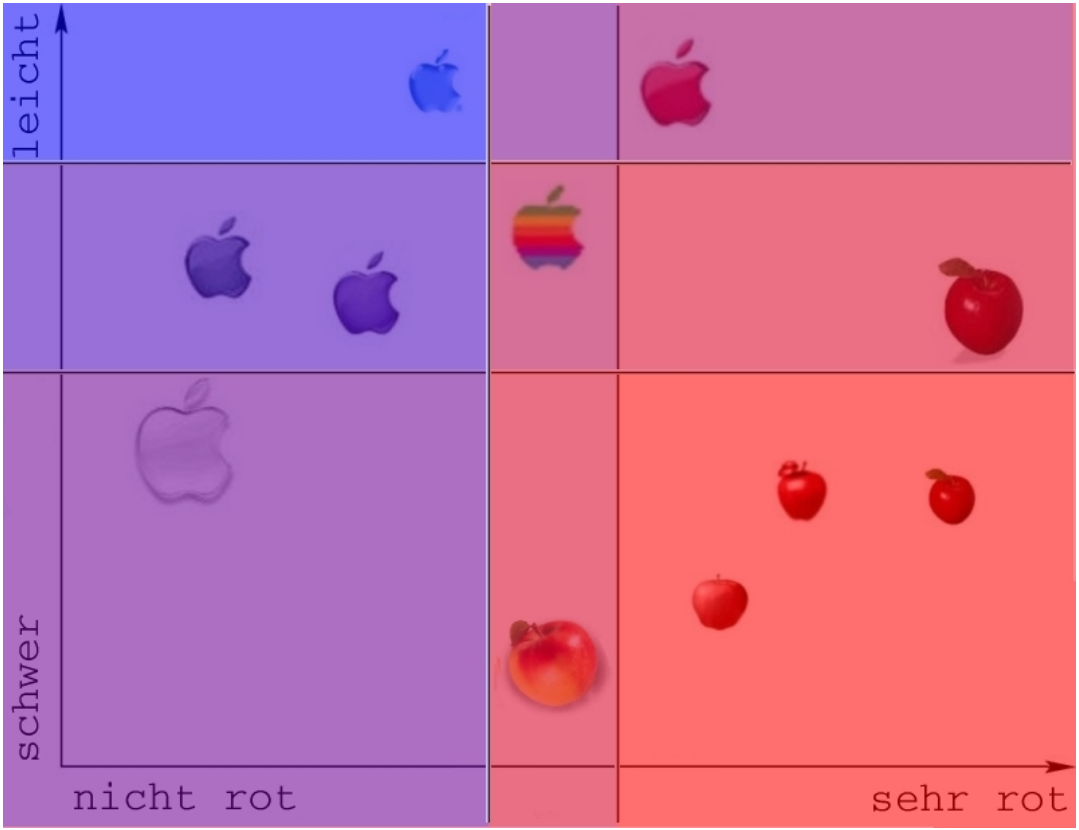
Boosting: 3nd Hypothesis



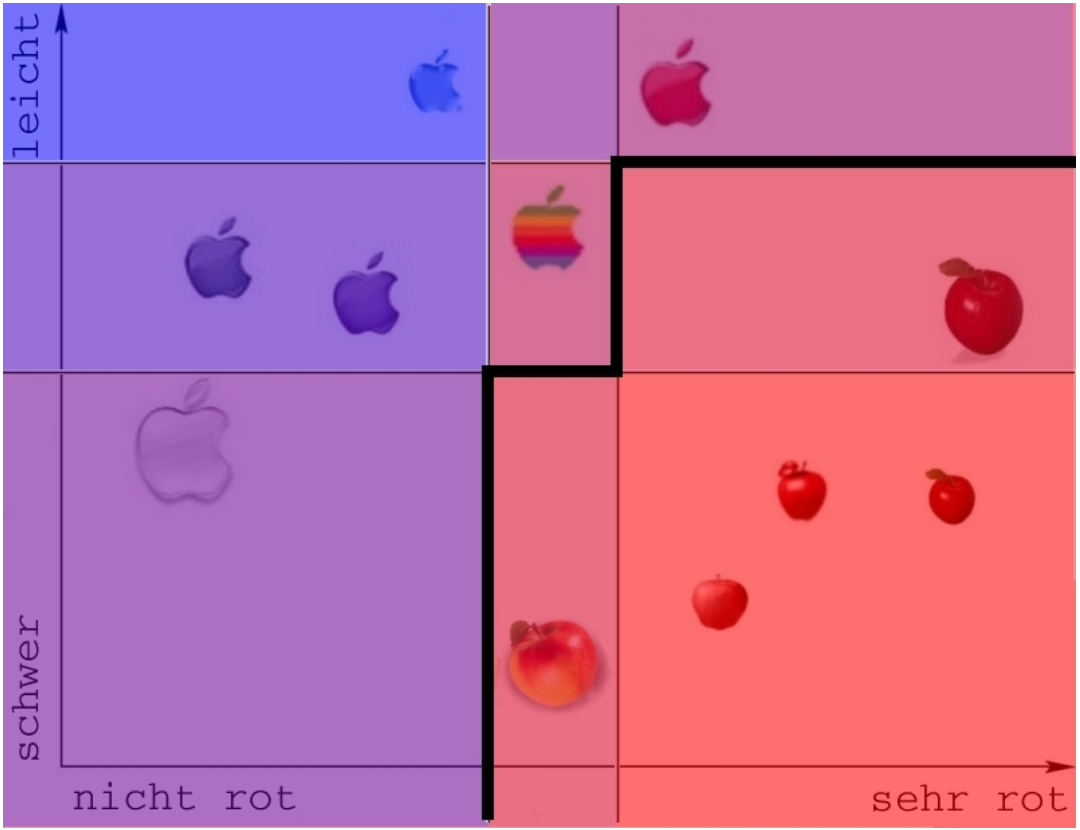
Boosting: 4th Hypothesis



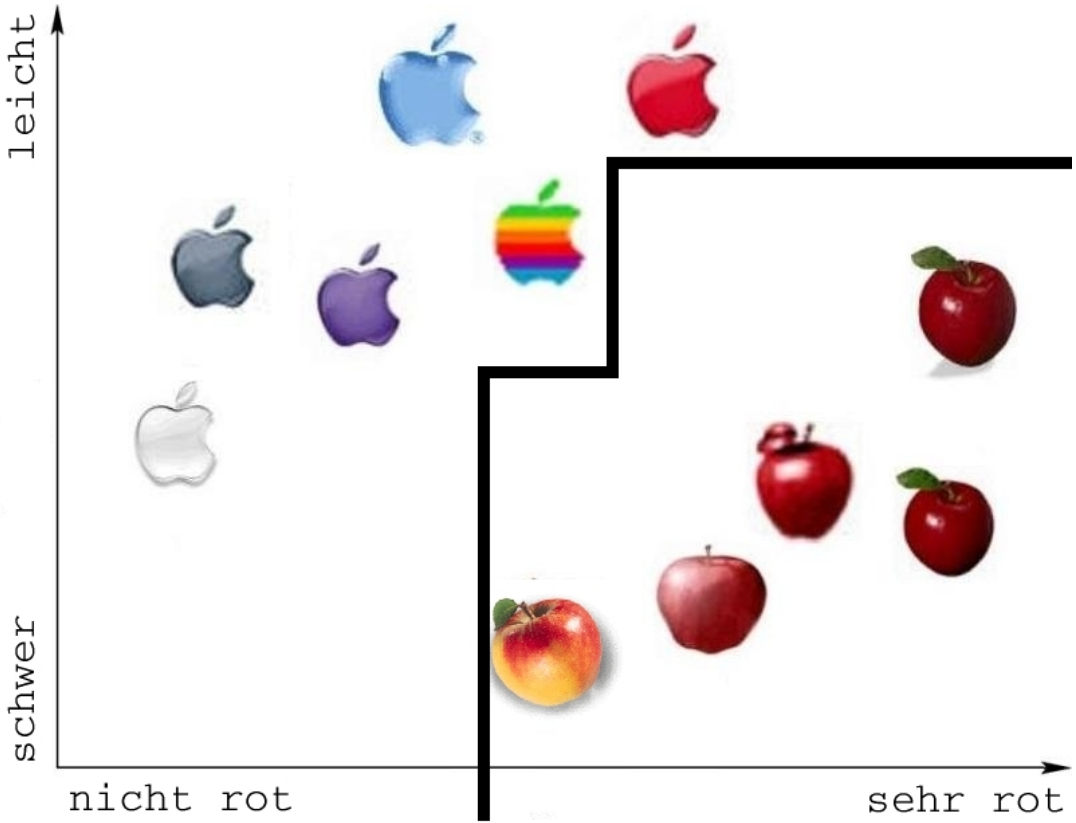
Combination of Hypotheses



Decision



Decision



Ensemble learning for classification

- Ensemble for binary classification consists of
  - Hypotheses (basis functions) $\{h_t(\mathbf{x}) : t = 1, \ldots, T\}$
    - of some hypothesis class $\mathcal{H} = \{h \mid h : \mathbf{x} \mapsto \pm 1\}$
  - Weights $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_T]$
    - satisfying $\alpha_t \geq 0$
- Classification Output: weighted majority of the votes
  - $f_{\text{Ens}}(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})$
- How to find the hypotheses and their weights?

  $+$ Boosting by Filtering (Schapire 1990)

  $+$ Boosting by Majority (Freund 1995)

  $-$ Bagging (Breiman 1996): $\alpha_t = 1/T$

  $+$ AdaBoost (Freund & Schapire 1996)

Early Algorithms

Boosting by Filtering (Schapire 1990)

- Run weak learner on differently filtered example sets
- Combine the weak hypotheses; **Complex**
- Requires knowledge on the performance of weak learner

Boosting by Majority (Freund 1995)

- Run weak learner on same, but differently weighted set
- Combine the weak hypotheses linearly – Majority vote
- Requires knowledge on the performance of weak learner

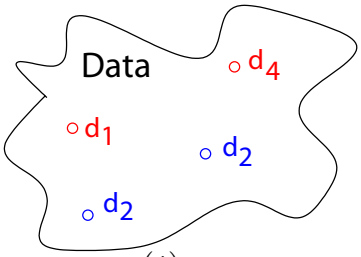
Bagging (Breiman 1996)

- Run learner on bootstrap replicates of the training set
- Average weak hypotheses – Majority vote
- Shown to reduce the variance (in regression)
- **not emphasizing "difficult" examples**

AdaBoost algorithm

Input: N examples $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$

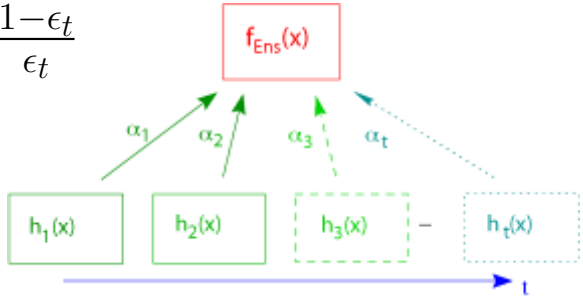
Initialize: $d_n^{(1)} = 1/N$ for all $n = 1 \ldots N$



Do for $t = 1, \ldots, T$,

1. Train base learner according to example distribution $\mathbf{d}^{(t)}$ and obtain hypothesis $h_t : \mathbf{x} \mapsto \{\pm 1\}$.
2. compute weighted error $\epsilon_t = \sum_{n=1}^{N} d_n^{(t)} \mathbf{I}(y_n \neq h_t(\mathbf{x}_n))$
3. compute hypothesis weight $\alpha_t = \frac{1}{2} \log \frac{1-\epsilon_t}{\epsilon_t}$
4. update example distribution

$$d_n^{(t+1)} = d_n^{(t)} \exp\left(-\alpha_t y_n h_t(\mathbf{x}_n)\right) / Z_t$$



Output: final hypothesis $f_{\text{Ens}}(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})$

Desirable Properties of the Learner

- generates “simple” hypotheses

  $\Rightarrow$ Uses small base hypothesis set $\mathcal{H}$
- Should (approximately) minimize weighted error

$$h_t = \operatorname*{argmin}_{h \in \mathcal{H}} \epsilon_t(h)$$

  where

$$\epsilon_t(h) = \sum_{n=1}^{N} d_n^{(t)} \mathbf{I}(y_n \neq h(\mathbf{x}_n))$$

- Often resampling techniques are used
  - Randomly draw subsets according to distribution $\mathbf{d}^{(t)}$
  - Run weak learner on the subset

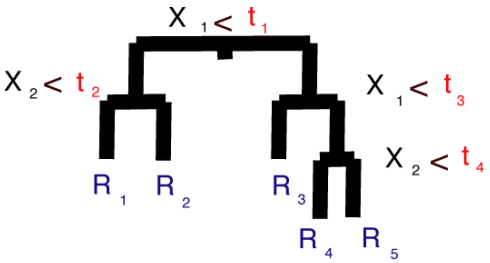
$\Rightarrow$ Allows usage of arbitrary weak learner

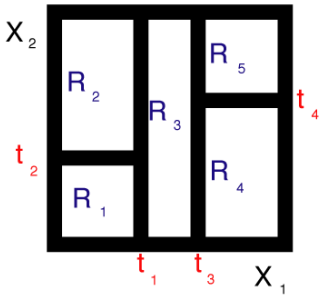
Weak Learners for Boosting

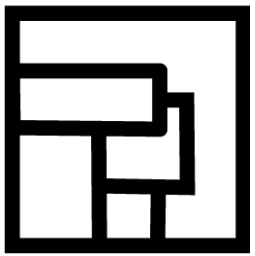
- Decision stumps (axis parallel splits)
- Decision trees (e.g. C4.5 by Quinlan 1996)
- Multi-layer Neural networks (e.g. for OCR)
- Radial basis function networks (e.g. UCI benchmarks, etc)

Decision trees:

- Hierarchical and recursive partitioning on the input space
- Many approaches, usually axis parallel splits






Impossible

Hypothesis weight $\alpha_t = \frac{1}{2}\log\frac{1-\epsilon_t}{\epsilon_t}$

. . . is equivalent to minimizing the following loss function in each iteration:

$$E(\alpha) = \sum_{n=1}^{N} d_n^{(t)} \exp(-\alpha y_n h_t(\mathbf{x}_n))$$

Excercise!

Use that $h_t(\mathbf{x}_n) \in \{-1, +1\}$ and $\epsilon_t = \sum_{n=1}^{N} d_n^{(t)} \mathbf{I}(y_n \neq h(\mathbf{x}_n))$

Hypothesis weight $\alpha_t = \frac{1}{2}\log\frac{1-\epsilon_t}{\epsilon_t}$

...is equivalent to minimizing the following loss function in each iteration:

$$E(\alpha) = \sum_{n=1}^{N} d_n^{(t)} \exp(-\alpha y_n h_t(\mathbf{x}_n))$$

$$\begin{aligned}
\frac{\mathrm{d}E(\alpha)}{\mathrm{d}\alpha} &= -\sum_{n=1}^{N} d_n^{(t)} y_n h_t(\mathbf{x}_n) \exp(-\alpha y_n h_t(\mathbf{x}_n)) && (2)\\
&= \sum_{y_n \neq h_t(\mathbf{x}_n)} d_n^{(t)} \exp(\alpha) - \sum_{y_n = h_t(\mathbf{x}_n)} d_n^{(t)} \exp(-\alpha) && (3)\\
&= \epsilon_t \exp(\alpha) - (1-\epsilon_t)\exp(-\alpha) = 0 && (4)
\end{aligned}$$

Hence,

$$\frac{\epsilon_t}{1-\epsilon_t} = \exp(-2\alpha) \qquad \Rightarrow \qquad \alpha = \frac{1}{2}\log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$$

Reweighting $d_n^{(t+1)} = d_n^{(t)} \exp\left(-\alpha_t y_n h_t(\mathbf{x}_n)\right) / Z_t$

Rewrite:

$$\frac{\mathrm{d}E(\alpha)}{\mathrm{d}\alpha} = -\sum_{n=1}^{N} d_n^{(t)} y_n h_t(\mathbf{x}_n) \exp(-\alpha y_n h_t(\mathbf{x}_n)) \tag{5}$$

$$= -\sum_{n=1}^{N} d_n^{(t+1)} y_n h_t(\mathbf{x}_n) \stackrel{!}{=} 0 \tag{6}$$

Hence, the weighted error of h_t w.r.t. $\mathbf{d}^{(t+1)}$ is $\frac{1}{2}$:

$$\sum_{n=1}^{N} d_n^{t+1} \mathbf{I}(y_n \neq h_t(\mathbf{x}_n)) = \frac{1}{2}$$

since

$$\mathbf{I}(y_n \neq h_t(\mathbf{x}_n)) = \frac{1}{2} + \frac{1}{2} y_n h_t(\mathbf{x}_n).$$

Reweighting $d_n^{(t+1)} = d_n^{(t)} \exp\left(-\alpha_t y_n h_t(\mathbf{x}_n)\right) / Z_t$

Note that:

$$\begin{aligned} d_n^{(t+1)} &= d_n^{(t)} \exp\left(-\alpha_t y_n h_t(\mathbf{x}_n)\right) / Z_t \\ &= d_n^{(t-1)} \exp\left(-\alpha_{t-1} y_n h_{t-1}(\mathbf{x}_n)\right) \exp\left(-\alpha_t y_n h_t(\mathbf{x}_n)\right) / (Z_t Z_{t-1}) \\ &= \frac{1}{N} \exp\left(-y_n \sum_{r=1}^{t} \alpha_r h_r(\mathbf{x}_n)\right) / \prod_{r=1}^{t} Z_r \\ &= \frac{1}{N} \exp\left(-y_n f_t(\mathbf{x}_n)\right) / \prod_{r=1}^{t} Z_r, \end{aligned}$$

where f_t is the combined hypothesis at step t:

$$f_t(\mathbf{x}) = \sum_{r=1}^{t} \alpha_r h_t(\mathbf{x})$$

3. AdaBoost's Behavior on the Sample



Hill Inlet, Whitsunday Island

Main Sources:

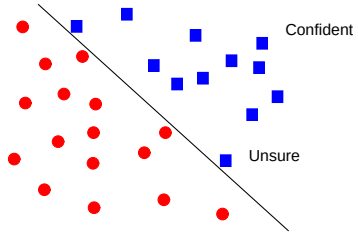
- Freund & Schapire 1996
- Schapire, Freund, Bartlett & Lee 1998

Margins and Classification

Motivation: Lose information in classifying a point as ± 1

Observation: Have higher confidence in correctly classified points, which are far from the decision boundary

Suggestion: Use real number to indicate confidence



Margin: $\rho_f(\mathbf{x}, y) = y f(\mathbf{x})$

Hyperplane: $\rho_{\mathbf{w},b}(\mathbf{x}, y) = y(\mathbf{w}^\top \mathbf{x} + b)$

Which Hyperplane?

- Function set used in boosting: Convex Hull of $\mathcal{H}$

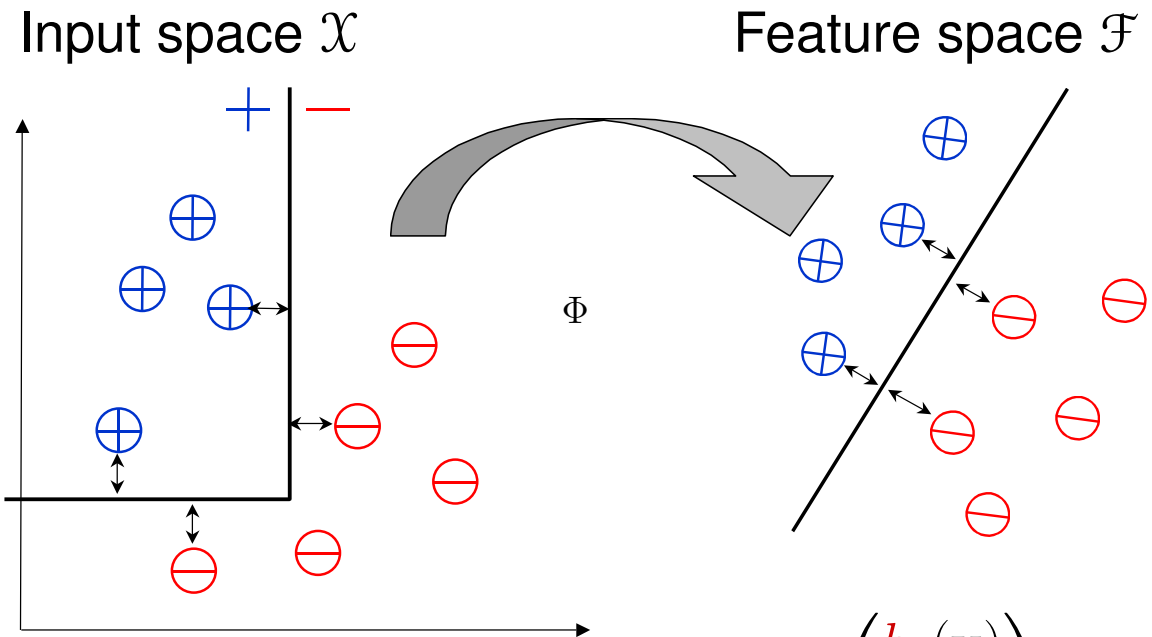
$$\mathcal{F} := \left\{ f : \mathbf{x} \mapsto \sum_{h \in \mathcal{H}} \alpha_h h(\mathbf{x}) \;\middle|\; \alpha_h \geq 0, \sum_{h \in \mathcal{H}} \alpha_h = 1 \right\}$$

- the α's are the parameters
- Find a hyperplane in the Feature Space spanned by the hypotheses set $\mathcal{H} = \{h_1, h_2, \ldots\}$
- Margin ρ for an example $(\mathbf{x}_i, y_n)$ and $f \in \mathcal{F}$:

$$\rho_n(\boldsymbol{\alpha}) := y_n f(\mathbf{x}_n) = y_i \sum_{t=1}^{T} \frac{\alpha_t}{\sum_t \alpha_t} h_t(\mathbf{x}_i)$$

- Margin ρ for a function $f \in \mathcal{F}$ by $\rho(\boldsymbol{\alpha}) := \min_{n=1,\ldots,N} \rho_n(\boldsymbol{\alpha})$

Large Margin and Linear Separation

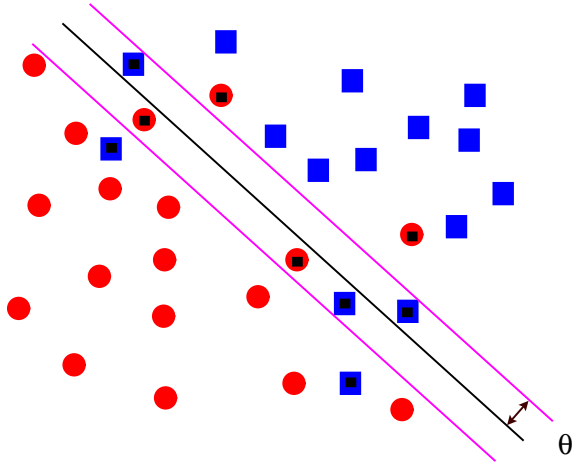


Linear separation in $\mathcal{F}$ is nonlinear separation in input space

$$\Phi(\mathbf{x}) = \begin{pmatrix} h_1(\mathbf{x}) \\ h_2(\mathbf{x}) \\ \vdots \end{pmatrix},$$

$$\mathcal{H} = \{h_1, h_2, \ldots\}$$

The Empirical Margin Error



Empirical margin error:

$$\hat{L}_N^{\theta}(f) = \frac{1}{N} \sum_{n=1}^{N} I[y_n f(x_n) \leq \theta]$$

(Scale of θ and f must match!)

The Empirical Margin Error

Lemma: Assume $h_t(x) \in \{-1, +1\}$ and

$$f_T(x) = \frac{\sum_{t=1}^{T} \alpha_t h_t(x)}{\sum_{t=1}^{T} \alpha_t} \qquad (f \in [-1, +1]).$$

Then

$$\hat{L}_N^{\theta}(f_T) = \frac{1}{N} \sum_{n=1}^{N} I[y_n f_T(x_n) \leq \theta]$$

$$\leq e^{\theta \sum_{t=1}^{T} \alpha_t} \left(\prod_{t=1}^{T} Z_t \right),$$

where Z_t is computed in AdaBoost:

$$Z_t = \sum_{n=1}^{N} d_n^t e^{-y_n \alpha_t h_t(\mathbf{x}_n)}$$

Proof

$$
\begin{aligned}
Z_t &= \sum_{n=1}^{N} d_n^t e^{-y_n \alpha_t h_t(x_n)} \\
&= \sum_{i: y_n = h_t(x_n)} d_n^t e^{-\alpha_t} + \sum_{i: y_n \neq h_t(x_n)} d_n^t e^{\alpha_t} \\
&= (1-\epsilon_t) e^{-\alpha_t} + \epsilon_t e^{\alpha_t}
\end{aligned}
$$

$$
\begin{aligned}
y f_T(x) \leq \theta \quad &\Rightarrow \quad y \sum_{t=1}^{T} \alpha_t h_t(x) \leq \theta \sum_{t=1}^{T} \alpha_t \\
&\Rightarrow \quad \exp\left(-y \sum_{t=1}^{T} \alpha_t h_t(x) + \theta \sum_{t=1}^{T} \alpha_t \right) \geq 1
\end{aligned}
$$

Hence, $$I[y f(x) \leq \theta] \leq \exp\left(-y \sum_{t=1}^{T} \alpha_t h_t(x) + \theta \sum_{t=1}^{T} \alpha_t \right)$$

Proof, Cont'd

$$d_n^{t+1} = d_n^t \frac{\exp\left(-\alpha_t y_n h_t(x_n)\right)}{Z_t}$$
$$= \frac{\exp\left(-\sum_{\tau=1}^{t} \alpha_\tau y_n h_\tau(x_n)\right)}{m \prod_\tau Z_\tau} \qquad \text{(Induction)}$$

$$\hat{\mathbf{E}}\{I[yf(x) \le \theta]\} \le \hat{\mathbf{E}}\left\{e^{-y\sum_{t=1}^{T}\alpha_t h_t(x) + \theta\sum_{t=1}^{T}\alpha_t}\right\}$$
$$= \frac{1}{N} e^{\theta\sum_{t=1}^{T}\alpha_t} \sum_{n=1}^{N} e^{-y_n\sum_{t=1}^{T}\alpha_t h_t(x_n)}$$
$$= e^{\theta\sum_{t=1}^{T}\alpha_t} \left(\prod_{t=1}^{T} Z_t\right) \underbrace{\sum_{n=1}^{N} d_n^{T+1}}_{=1}$$

Empirical Margin Error Bound

Claim: Selecting $\alpha_t = (1/2)\ln((1-\epsilon_t)/\epsilon_t)$ leads to the bound

$$Z_t = 2\sqrt{\epsilon_t(1-\epsilon_t)}.$$

Proof: Straightforward by substitution (α_t is minimizer of $Z_t(\alpha_t)$)

Conclusion: Recall $\hat{L}_N^\theta(f) = \frac{1}{N}\sum_{n=1}^N I[y_n f(x_n) \leq \theta]$

$$\hat{L}_N^\theta(f_T) \leq \prod_{t=1}^T \sqrt{4\epsilon_t^{1-\theta}(1-\epsilon_t)^{1+\theta}}$$

If $\epsilon_t = 1/2 - \gamma_t/2, \quad \gamma_t \geq \theta \; \forall t$

Then $\hat{L}_N^\theta(f_T) \to 0$ exponentially fast!

Training Error

Consider $\theta = 0$: $\hat{L}_N(f) = \frac{1}{N} \sum_{n=1}^{N} I[y_n f(x_n) \leq 0]$

Using $\ln x \leq x - 1$

$$\hat{L}_N(f_T) \leq \prod_{t=1}^{T} \sqrt{(1 - 4\gamma_t^2)}$$

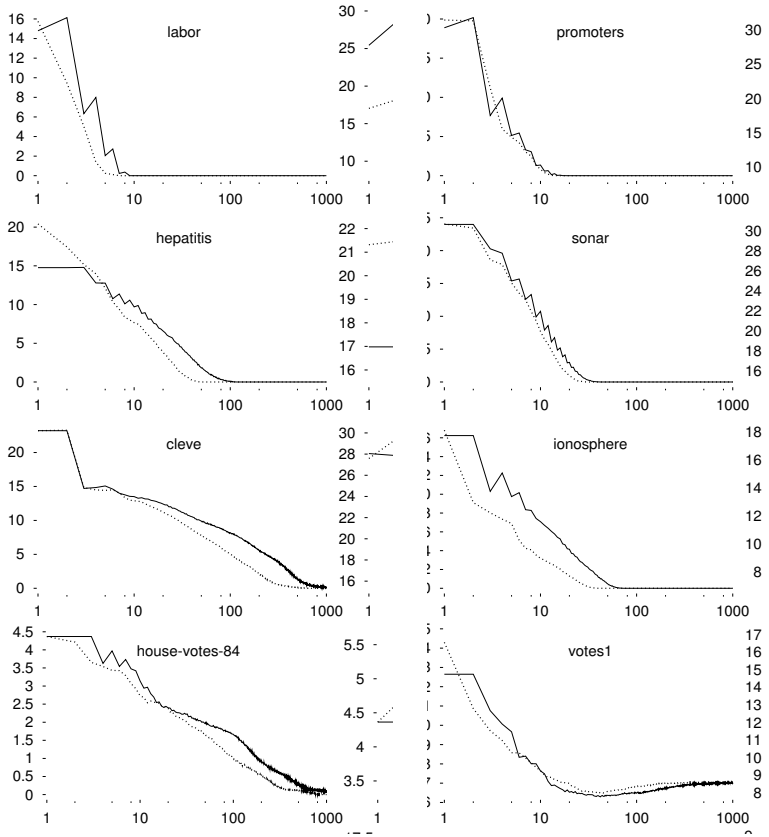
$$\leq e^{-2\sum_{t=1}^{T} \gamma_t^2} \to 0 \quad \text{if} \sum_{t=1}^{T} \gamma_t^2 \to \infty$$

Error can be driven to zero, if weak learner are sufficiently strong

If $\gamma_t \geq \gamma$ $(\forall t)$, then training error is zero after

$$T \leq \left\lceil \frac{2 \log N}{\gamma^2} \right\rceil$$

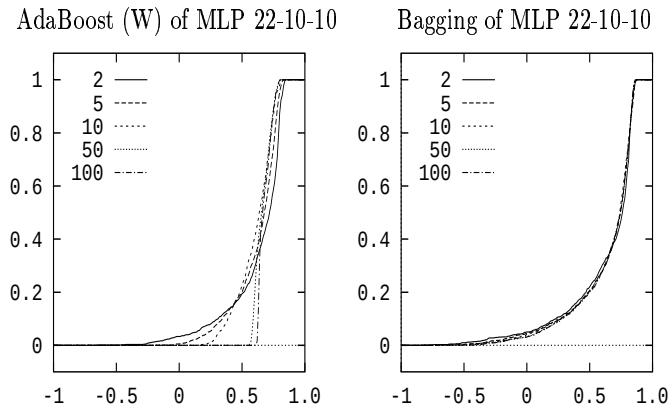
Training Error for Real Data



Source: Schapire and Singer 1999

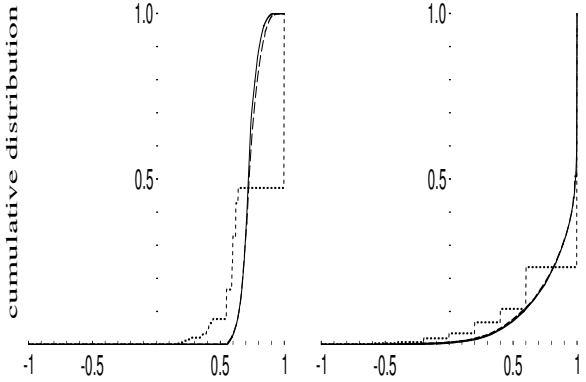
Margin Plots I

Margin plots: Histogram of $y_n f_T(x_n)$, $n = 1, 2, \ldots, N$



Neural Networks

Schwenk and Bengio 2000



Decison Trees

Schapire *et al.* 1998

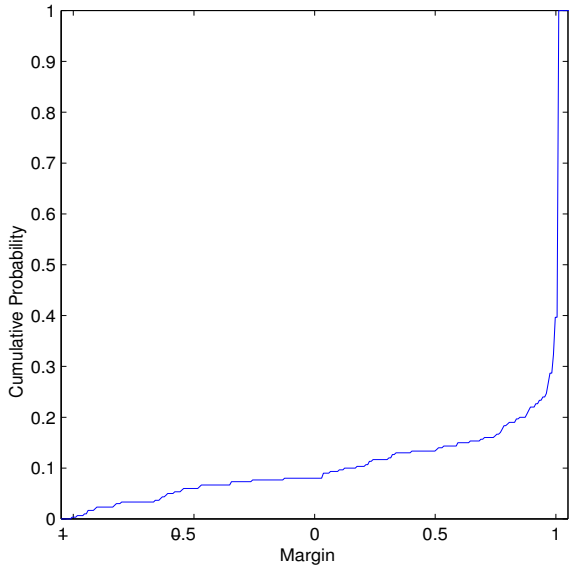
Observation:

- AdaBoost increases the margins of most points
- Bagging has a broader spectrum of distributions

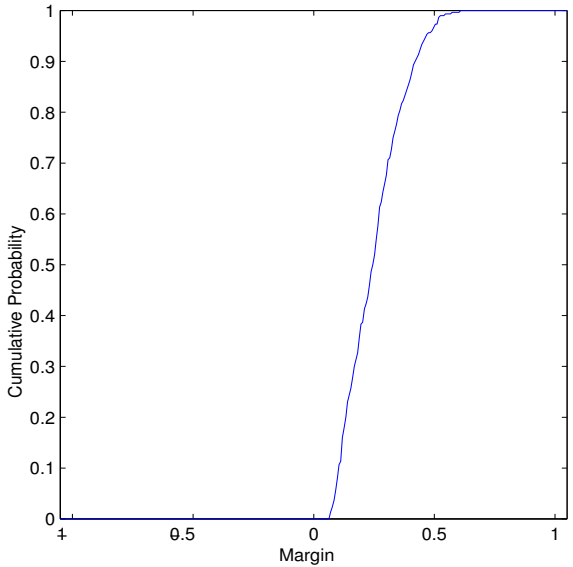
Margin Plots II

AdaBoost tends to increase small margins, while decreasing large margins

Bagging



AdaBoost



Source: Schapire et al. 1998, Rätsch et al., 1998

Contents of this Course

- Part I: Introduction to Boosting
  1. Basic Issues, PAC Learning
  2. The Idea of Boosting
  3. AdaBoost's Behavior on the Sample
- Part II: Some Theory
  4. Generalization Performance
  5. Boosting and Large Margins
  6. Leveraging as Greedy Optimization
- Part III: Extensions and Applications
  7. Boosting vs SVMs
  8. Improving Robustness & Extensions
  9. Applications

Slides available from the wiki and from

http://www.fml.mpg.de/~raetsch/boostintro

4. Generalization Performance

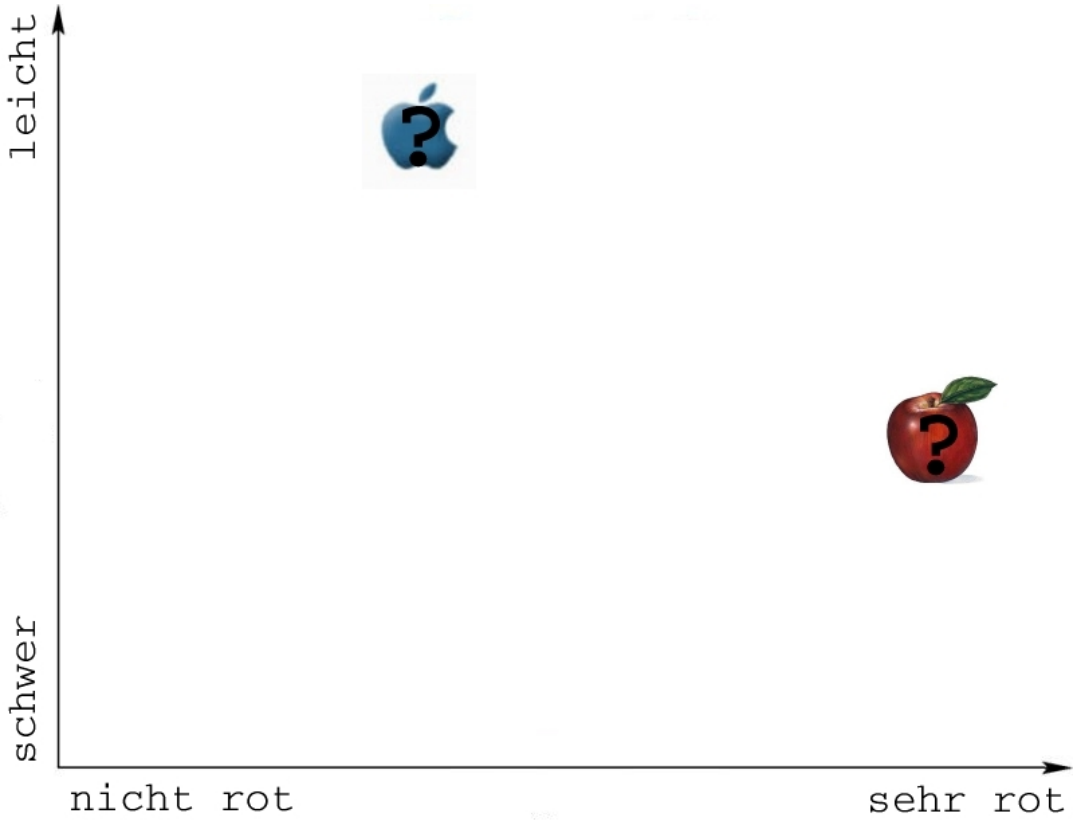


Hill Inlet, Whitsunday Island

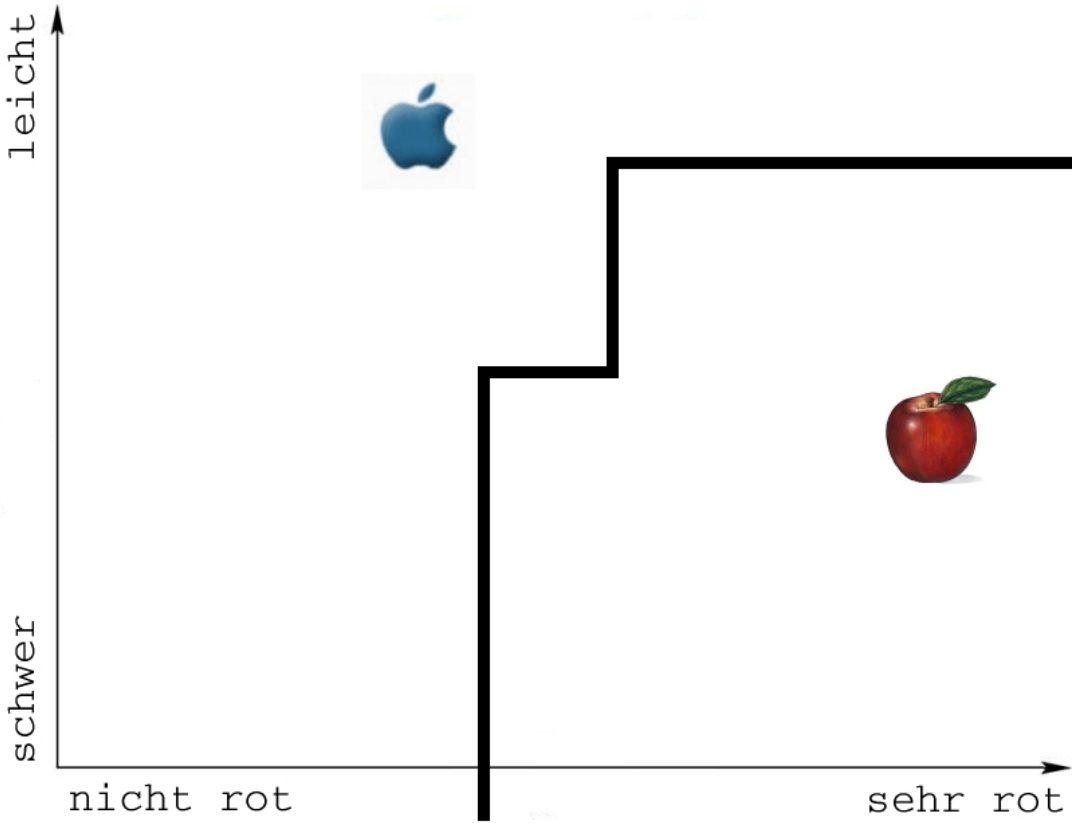
Main Sources:

- Freund & Schapire 1996
- Schapire, Freund, Bartlett & Lee 1998

Unseen Examples



Unseen Examples



Setup

Expected error:	$L(f) = \mathbf{E}\{I[yf(x) \leq 0]\}$
Data:	$(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$
Empirical error:	$\hat{L}_N(f) = \frac{1}{N} \sum_{n=1}^{N} I[y_n f(x_n) \leq 0]$
Empirical margin error:	$\hat{L}_N^{\theta}(f) = \frac{1}{N} \sum_{i=1}^{N} I[y_n f(x_n) \leq \theta]$
Empirical classifier:	$\hat{f}_N$

Main issue:	Bound $L(\hat{f}_N)$ in terms of $\hat{L}_N(\hat{f}_N)$
Standard VC bounds:	Use $\hat{L}_N(\hat{f}_N)$
Margin bounds:	Use $\hat{L}_N^{\theta}(\hat{f}_N)$

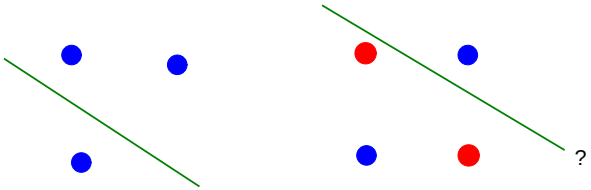
VC Dimension

Given: $\mathcal{F} = \{f : \mathbf{R}^d \mapsto \{-1, +1\}\}$

Question: How complex is the class?

Shattering: $\mathcal{F}$ shatters a set $X \subset \mathbf{R}^d$, if $\mathcal{F}$ achieves all dichotomies on X

VC-dimension: The size of the largest shattered subset of X



All 8 dichotomies

14 dichotomies



6 dichotomies

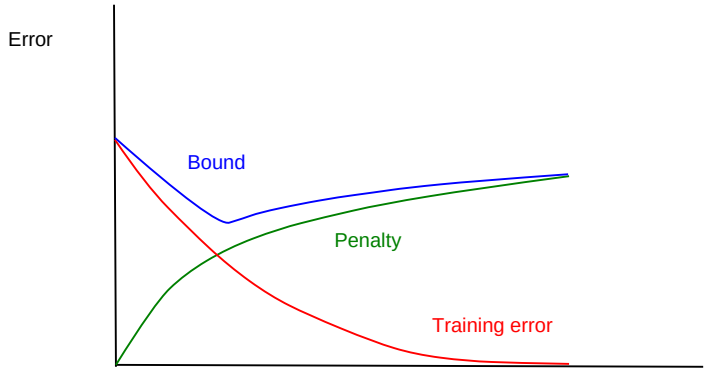
$$d_{\mathcal{F}} = 3$$

VC Bounds

Recall

$$L(f) \leq \hat{L}_N(f) + c_1\sqrt{\frac{d_{\mathcal{F}}}{N}} + c_2\sqrt{\frac{\log\frac{1}{\delta}}{N}}$$

Tightness:	Bound becomes tight for increasing sample size
Empirical Error:	Decreases with complexity of $\mathcal{F}$
Confidence term:	Increases with complexity classes $\mathcal{F}$
Overfitting:	Occurs when $d_{\mathcal{F}}$ becomes very large



VC Bounds for Convex Combination

Recall

$$f_T(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$$

Let

$$\mathrm{co}_T(\mathcal{H}) = \left\{ f : \; f(x) = \sum_{t=1}^{T} \alpha_t h_t(x), \; \alpha_n \geq 0, \; \sum_{t=1}^{T} \alpha_t = 1 \right\}$$

For any $f \in \mathcal{F} := \mathrm{co}_T(\mathcal{H})$, with probability $1 - \delta$

$$L(f) \leq \hat{L}_N(f) + c_1 \sqrt{\frac{d(\mathrm{co}_T(\mathcal{H}))}{N}} + c_2 \sqrt{\frac{\log \frac{1}{\delta}}{N}}$$

Problem with VC Bound: VC-dimension of $\mathrm{co}_T(\mathcal{H})$ may be large, often

$$d(\mathrm{co}_T(\mathcal{H})) \approx T \log(T) \; d(\mathcal{H})$$

But sufficient to get original Boosting results . . .

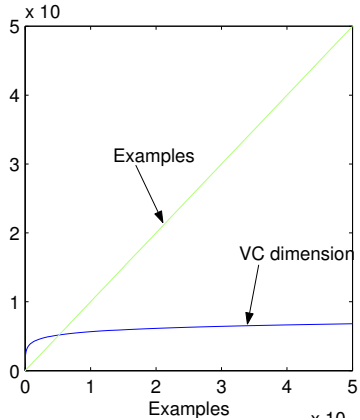
PAC Boosting – VC dim. of comb. hypothesis

- AdaBoost needs at most $\lceil \frac{2 \log N}{\gamma^2} \rceil$ iterations to obtain consistent hypothesis
- Let d be the VC dimension of the hypothesis class $\mathcal{H}$.
- Then the VC dimension of the class of AdaBoost's combined functions is bounded by

$$d_{Ens}(N, \gamma) = \mathcal{O}\Big(d \underbrace{\frac{\log(N)}{\gamma^2}}_{\sim T} \log\left(\frac{\log(N)}{\gamma^2}\right)\Big) = \mathcal{O}\left(d \log(N) \log^2(N)\right).$$

An Example

- VC dimension $d = 2$ (e.g. decision stumps)
- $\epsilon_t \leq 0.4 = \frac{1}{2} - \frac{1}{2}\gamma$
  $\Rightarrow \gamma \geq 0.2$



PAC Boosting – putting things together

Assume weak learner is strong enough (weak PAC learner):

- AdaBoost generates consistent hypothesis: $\hat{L}_N(f) = 0$
- VC dimension of a consistent combined hypotheses class

$$d_N \leq \mathcal{O}(d \log(N) \log^2(N))$$

- With probability $1 - \delta$ holds

$$L(f) \leq 0 + c_1 \sqrt{\frac{d_N}{N}} + c_2 \sqrt{\frac{\log \frac{1}{\delta}}{N}}$$

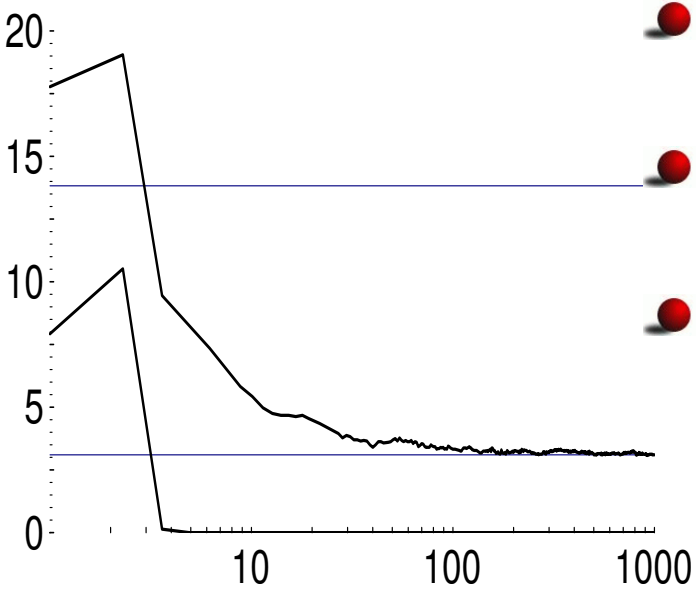
- The **generalization** error "$0 + \varepsilon(N)$" can be made arbitrarily small by increasing the sample size N.

PAC Booosting – Digestion

- properties of weak learner imply exponential convergence to a consistent hypothesis
- Fast convergence ensures small VC dimension of the combined hypothesis
- small VC implies small deviation from the empirical risk
- for any $\varepsilon > 0$ and $\delta > 0$ exists a sample size N, such that with probability $1 - \delta$ the expected risk is smaller than ε
- Any weak learner can be boosted to achieve an arbitrary high accuracy! ($\rightsquigarrow$ strong learner)

A “strange” Phenomenon

boosting C4.5 on “letter” data



- test error does not increase
  ⇝ even after 1000 iterations!
- it continues to drop
  ⇝ even after training error is 0!
- Occam’s razor predicts simpler rule is better
  ⇝ wrong in this case!?

Needs a better explanation!

Margin Bounds (Schapire, Freund, Bartlett & Lee 1998)

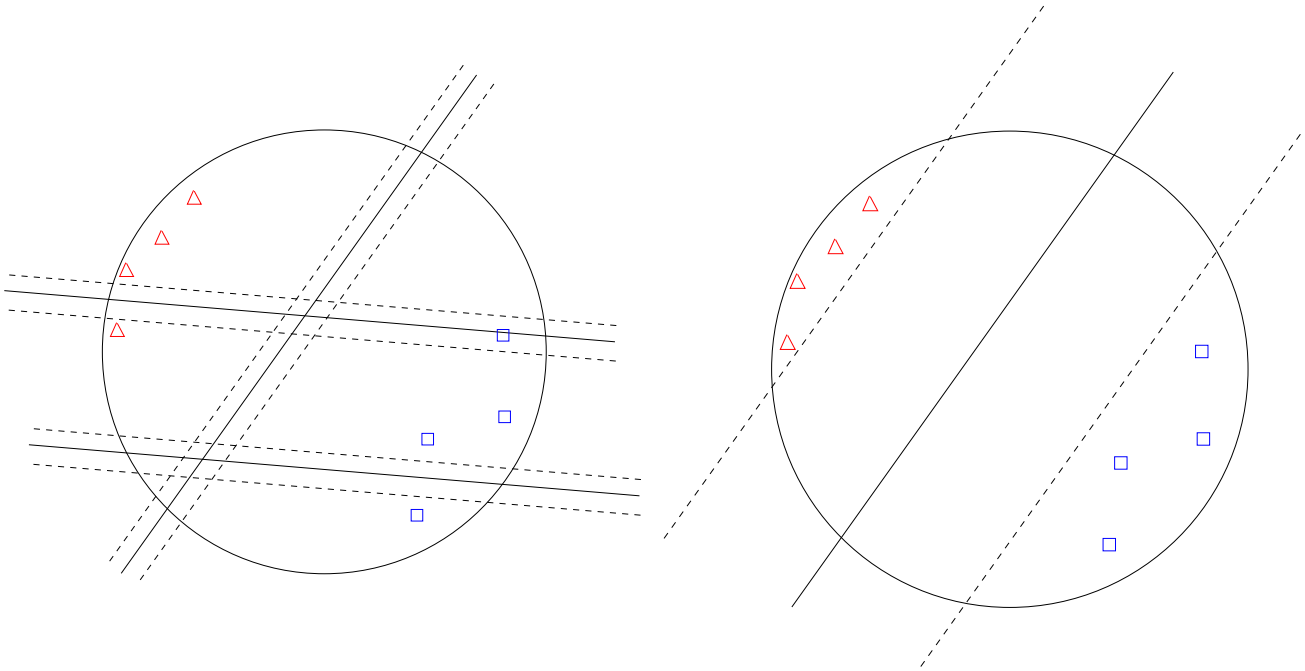
For any $f \in \mathrm{co}_T(\mathcal{H})$ and $\theta \in (0,1]$, with probability $1-\delta$

$$L(f) \leq \hat{L}_N^\theta(f) + \frac{c_1}{\theta}\sqrt{\frac{d(\mathcal{H})}{N}} + c_2\sqrt{\frac{\log\frac{1}{\delta}}{N}}$$

Observe:

- ⋆ No dependence on T - is overfitting absent?
- ⋆ Luckiness incorporated through $\hat{L}_N^\theta(f)$
- ⋆ Limit $\theta \to 0$ diverges
- ⋆ Much better that VC bounds (if lucky)
- ⋆ Overfitting absent (if lucky) - no T-dependence

Large Margin $\Rightarrow$ low Complexity? – Why?



An Introduction to Boosting
Part III

Gunnar Rätsch

Friedrich Miescher Laboratory
of the Max Planck Society

http://www.fml.tuebingen.mpg.de/~raetsch

Help with earlier presentations: Ron Meir, Klaus-R. Müller

Contents of this Course

- Part I: Introduction to Boosting
  1. Basic Issues, PAC Learning
  2. The Idea of Boosting
  3. AdaBoost's Behavior on the Sample
- Part II: Some Theory
  4. Generalization Error Bounds
  5. Boosting and Large Margins
  6. Leveraging as Greedy Optimization
- Part III: Extensions and Applications
  7. Boosting vs. SVMs
  8. Improving Robustness & Extensions
  9. Applications

Slides available from the wiki and from

`http://www.fml.mpg.de/~raetsch/boostintro`

7. Boosting vs. SVMs



Hill Inlet, Whitsunday Island

Main Sources:

- Schapire, Freund, Bartlett & Lee 1998
- Rätsch et al. 2000, 2001

SVM's Mathematical Program

The SVM minimization of

$$\begin{aligned} \min_{\mathbf{w}\in\mathcal{F}_\Phi} \quad & \tfrac{1}{2}\|\mathbf{w}\|^2 \\ \text{subject to} \quad & y_n\langle \mathbf{w}, \Phi(\mathbf{x}_n)\rangle \geq 1, \quad n = 1,\ldots,N. \end{aligned} \tag{1}$$

reformulate as maximization of the margin ρ

$$\begin{aligned} \max_{\mathbf{w}\in\mathcal{F}_\Phi,\rho\in\mathbf{R}_+} \quad & \rho \\ \text{subject to} \quad & y_n \sum_{j=1}^{D} w_j \Phi_j(\mathbf{x}_n) \geq \rho \quad \text{for} \quad n = 1,\ldots,N \\ & \|\mathbf{w}\|_2 = 1, \end{aligned} \tag{2}$$

where $D = \dim(\mathcal{F})$ and Φ_j is the j-th component of Φ in feature space:

$$\Phi_j = P_j[\Phi]$$

Boosting's Mathematical Program

- master hypothesis

$$f(\mathbf{x}) = \sum_{t=1}^{T} \frac{\alpha_t}{\|\boldsymbol{\alpha}\|_1} h_t(\mathbf{x})$$

- base hypotheses h_t produced by the base algorithm.

AdaBoost* & Arc-GV solutions are (asymptotically) the same as linear program solution, maximizing the smallest margin ρ:

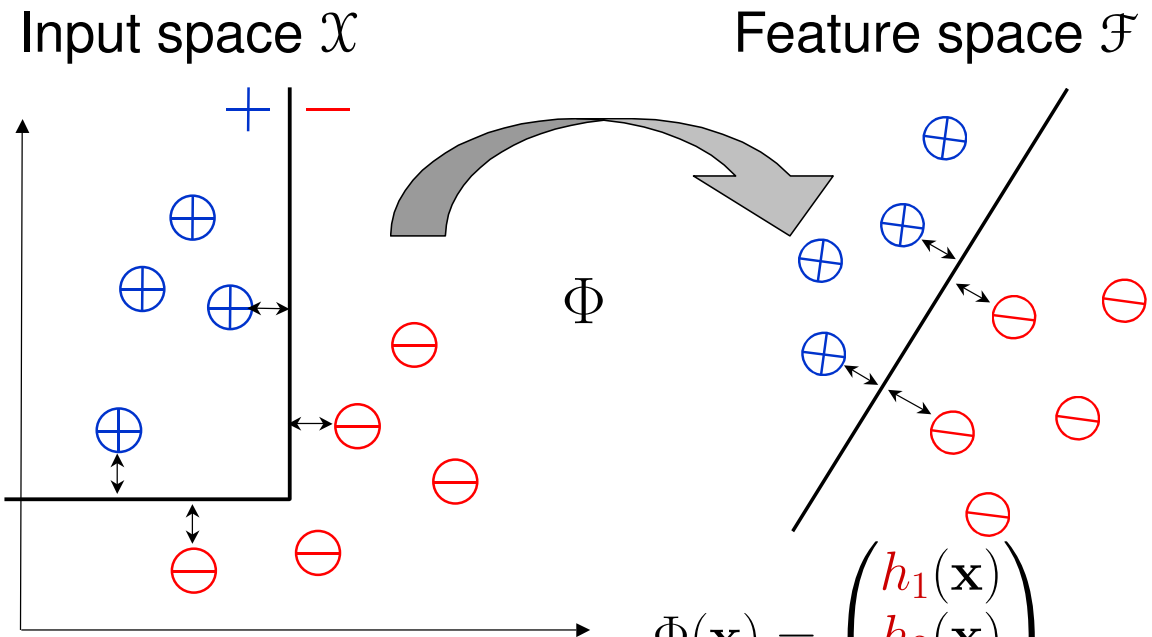
$$\max_{\mathbf{w}\in\mathbf{R}^J, \rho\in\mathbf{R}_+} \quad \rho$$

$$\text{subject to} \quad y_n \sum_{j=1}^{J} w_j h_j(\mathbf{x}_n) \geq \rho \quad \text{for} \quad n = 1, \ldots, N \qquad (3)$$

$$\|\mathbf{w}\|_1 = 1,$$

where J is the number of hypotheses in $\mathcal{H}$.

Large Margin and Linear Separation



Linear separation in $\mathcal{F}$ is nonlinear separation in input space

$$\Phi(\mathbf{x}) = \begin{pmatrix} h_1(\mathbf{x}) \\ h_2(\mathbf{x}) \\ \vdots \end{pmatrix},$$

$$\mathcal{H} = \{h_1, h_2, \ldots\}$$

$$k(\mathbf{x}, \mathbf{y}) = \Phi(\mathbf{x}) \cdot \Phi(\mathbf{y})$$

Even more explicit

Observe that any hypothesis set $\mathcal{H}$ implies a mapping Φ by

$$\Phi : \mathbf{x} \mapsto [h_1(\mathbf{x}), \ldots, h_J(\mathbf{x})]^\top ,$$

and a kernel

$$k(\mathbf{x}, \mathbf{y}) = (\Phi(y) \cdot \Phi(\mathbf{y})) = \sum_{j=1}^{J} h_j(\mathbf{x}) h_j(\mathbf{y})$$

which could in principle be used for SVM learning.

$\Rightarrow$ Any hypothesis set $\mathcal{H}$ spans a feature space $\mathcal{F}$.

$\Rightarrow$ For any feature space $\mathcal{F}$ (spanned by Φ), the corresponding hypothesis set $\mathcal{H}$ can be readily constructed by

$$h_j = P_j[\Phi].$$

About Norms

- ℓ_2–norm (SVMs) $\Rightarrow$ Euclidean distance between hyperplane and examples is maximized
- using an arbitrary ℓ_p–norm constraint on the weight vector leads to maximizing the ℓ_q–distance between hyperplane and training points, where $\frac{1}{q} + \frac{1}{p} = 1$.
- Boosting-type algorithms maximize minimum ℓ_∞–distance of the training points to the hyperplane

What happens in the long run?

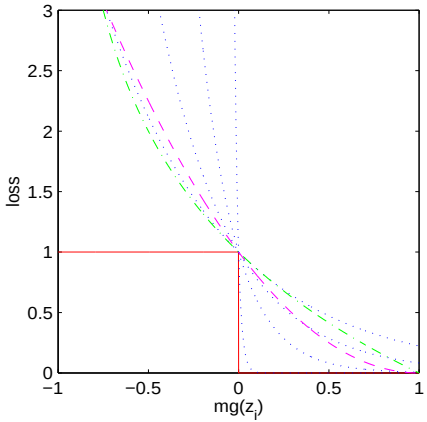
- Explicit expression for $d_n^{(t+1)}$:

$$d_n^{(t+1)} = \frac{\exp\left\{-\rho_n(\boldsymbol{\alpha}^{(t)})\right\}^{\|\boldsymbol{\alpha}^{(t)}\|}}{\sum_{n'=1}^{N} \exp\left\{-\rho_{n'}(\boldsymbol{\alpha}^{(t)})\right\}^{\|\boldsymbol{\alpha}^{(t)}\|}}$$

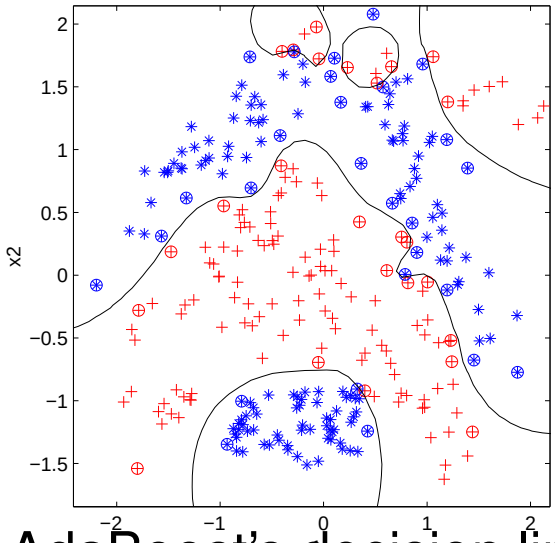
  $\rightsquigarrow$ **Soft-Max Function** with parameter $\|\boldsymbol{\alpha}^{(t)}\|_1$

- $\|\boldsymbol{\alpha}\|_1$ will increase monotonically ($\sim$ linear)

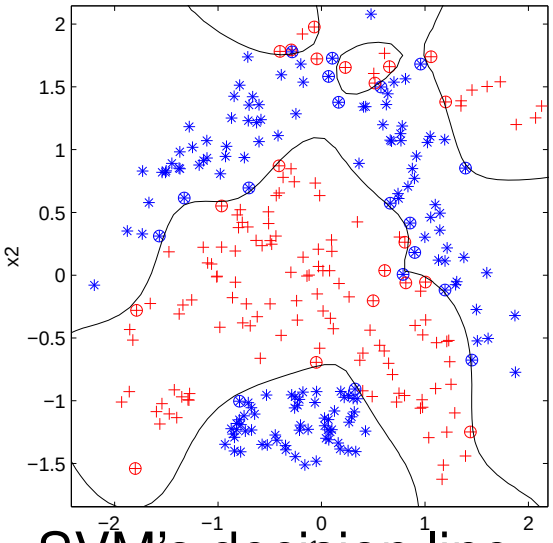
$\rightsquigarrow$ the d's concentrate on a **few** difficult patterns
$\rightarrow$ **"Support Patters"**



Support Patterns vs. Support Vectors



AdaBoost's decision line



SVM's decision line

These decision lines are for a low noise case with similar generalization errors. In AdaBoost, RBF networks with 13 centers were used.

Boosting vs. SVMs I

- SVMs

$$R[f] \leq R_{emp}[f] + \mathcal{O}\left(\sqrt{\frac{\log\left(N\theta^2\right)}{\theta^2 N} + \frac{\log(1/\eta)}{N}}\right).$$

- Boosting

$$R[f] \leq R^{\theta}_{emp}[f] + \mathcal{O}\left(\sqrt{\frac{d\log^2\left(\frac{N}{d}\right)}{\theta^2 N} + \frac{\log(1/\delta)}{N}}\right)$$

- **independent of the dimensionality of the space!**

Representer Theorem for Kernel Algorithms

Let c be an arbitrary cost function and r be a strictly monotonically increasing function on $[0, \infty)$ (for instance $r(\|\mathbf{w}\|_2) = \lambda \|\mathbf{w}\|_2$, $\lambda > 0$).

$$\min_{\mathbf{w} \in \mathcal{F}_\Phi} \sum_{n=1}^{N} c(y_n, \underbrace{f_{\mathbf{w}}(\mathbf{x}_n)}_{\langle \mathbf{w}, \Phi(\mathbf{x}_n) \rangle}) + r(\|\mathbf{w}\|_2) \qquad (4)$$

Then for any N examples $(\mathbf{x}_1, y_1), \ldots, (x_N, y_N)$ and any mapping Φ, then **any solution** $\mathbf{w}^*$ of $(??)$ can be expressed by

$$\mathbf{w}^* = \sum_{n=1}^{N} \alpha_n \Phi(\mathbf{x}_n)$$

for some $\alpha_n \in \mathbf{R}$, $n = 1, \ldots, N$.

Only ℓ_2-norm regularizers have this property!

Representer Theorem for Boosting Algs

Let c be an arbitrary cost function and r be a monotonically increasing function on $[0, \infty)$.

$$\min_{\mathbf{w} \in \mathcal{F}_{\mathcal{H}}} \sum_{n=1}^{N} c(y_n, \underbrace{f_{\mathbf{w}}(\mathbf{x}_n)}_{\sum_{h \in \mathcal{H}} w_h h(\mathbf{x}_n)}) + r(\|\mathbf{w}\|_1) \tag{5}$$

Then for any N example $(\mathbf{x}_1, y_1), \ldots, (x_N, y_N)$ and any compact hypothesis set $\mathcal{H}$, there **exist solutions** $f_{\mathbf{w}^*}$ of $(??)$ that can be expressed by

$$f_{\mathbf{w}^*}(\mathbf{x}) = \sum_{n=1}^{N} \alpha_n h_n(\mathbf{x}).$$

Most of the w's can be made zero!

The ℓ_1-norm is the only convex regularizer that has this property!

Boosting vs. SVMs II

- Boosting, in contrast to SVMs, performs the computation explicitly in feature space.
- using the ℓ_1–norm instead of the ℓ_2–norm this becomes feasible $\rightarrow$ sparse solutions
- few hypotheses/dimensions $h_j = P_j[\Phi]$ needed to express the solution
- Boosting considers only the most important dimensions in feature space (at most N)

Ingredients of Boosting

- combinations of “weak hypotheses”
- large margins (for two-class classification)
- algorithms for solving convex optimization problems
  - re-weighting is optimization trick
  - Barrier minimization (exponential) or column generation
- sparseness in “Feature Space” allows efficient algorithms

6. Boosting as Greedy Optimization



Hill Inlet, Whitsunday Island

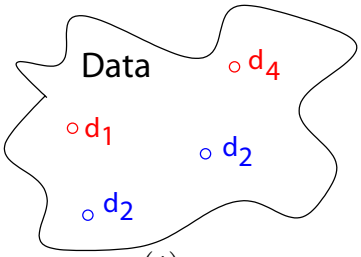
Main Sources:

- Kivinen & Warmuth 1999, Lafferty 1999
- Collins, Schapire & Singer 2002
- Rätsch, Mika & Warmuth 2001, Rätsch 2001

AdaBoost algorithm

Input: N examples $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$

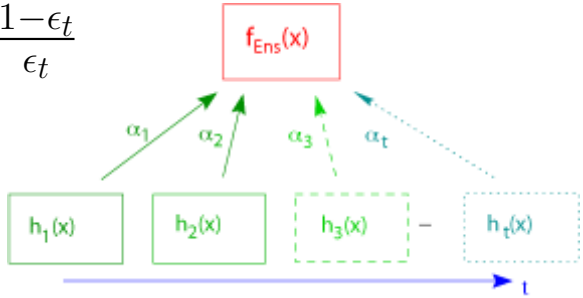
Initialize: $d_n^{(1)} = 1/N$ for all $n = 1 \ldots N$



Do for $t = 1, \ldots, T$,

1. Train base learner according to example distribution $\mathbf{d}^{(t)}$ and obtain hypothesis $h_t : \mathbf{x} \mapsto \{\pm 1\}$.
2. compute weighted error $\epsilon_t = \sum_{n=1}^{N} d_n^{(t)} \mathbf{I}(y_n \neq h_t(\mathbf{x}_n))$
3. compute hypothesis weight $\alpha_t = \frac{1}{2} \log \frac{1-\epsilon_t}{\epsilon_t}$
4. update example distribution

$$d_n^{(t+1)} = d_n^{(t)} \exp\left(-\alpha_t y_n h_t(\mathbf{x}_n)\right) / Z_t$$



Output: final hypothesis $f_{\text{Ens}}(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})$

Loss Function for AdaBoost

- AdaBoost stepwise minimizes a function

$$\mathrm{G}[f_{\boldsymbol{\alpha}}] = \sum_{n=1}^{N} \exp\left\{-y_n f_{\boldsymbol{\alpha}}(\mathbf{x}_n)\right\}$$

- The gradient of $\mathrm{G}[f_{\boldsymbol{\alpha}^{(t)}}]$ gives the example weights used for AdaBoost:

$$\frac{\partial\, \mathrm{G}[f_{\boldsymbol{\alpha}^{(t)}}]}{\partial f(\mathbf{x}_n)} = \exp\left\{-y_n f_{\boldsymbol{\alpha}^{(t)}}(\mathbf{x}_n)\right\} =: d_n^{(t+1)}$$

- The hypothesis coefficient α_t is chosen, such that $\mathrm{G}[f_{\boldsymbol{\alpha}^{(t)}}] = \mathrm{G}[f_{\boldsymbol{\alpha}^{(t-1)}} + \alpha_t h_t]$ is minimized:

$$\alpha_t = \operatorname*{argmin}_{\alpha_t \in \mathbf{R}} \mathrm{G}[f_{\boldsymbol{\alpha}^{(t)}}] = \frac{1}{2} \log \frac{1-\epsilon_t}{\epsilon_t}$$

The Generic Leveraging Algorithm

Input: loss $\mathrm{G} : \mathbf{R}^N \to \mathbf{R}$, N examples $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$
Initialize: $f_0 \equiv 0$, $\mathbf{d}^{(1)} = \nabla \mathrm{G}[f_0]$ (often $\sim 1/N$)
Do for $t = 1, \ldots, T$,

1. Run weak learner according to distribution $\mathbf{d}^{(t)}$ and obtain hypothesis h_t.
2. Compute hypothesis weight α_t such that $\mathrm{G}[f_{t-1} + \alpha_t h_t]$ is minimized
3. Update the combined hypothesis: $f_t = f_{t-1} + \alpha_t h_t$
4. Update example distribution: $\mathbf{d}^{t+1} = \nabla \mathrm{G}[f_t]$

Output: final hypothesis $f_T(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})$

Leveraging = Boosting without PAC Boosting property

Examples

- AdaBoost (Freund & Schapire, 94) & Confidence Rated Boosting

$$\mathrm{G}[f_{\boldsymbol{\alpha}}] = \sum_{n=1}^{N} \exp\left\{-y_n f_{\boldsymbol{\alpha}}(\mathbf{x}_n)\right\} \qquad d_n = \exp(-y_n f_{\boldsymbol{\alpha}}(\mathbf{x}_n))$$

- Logistic Regression Boosting (Friedman et al., Collins et al., 2000)

$$\mathrm{G}[f_{\boldsymbol{\alpha}}] = \sum_{n=1}^{N} \log\left(1 + \exp\left\{-y_n f_{\boldsymbol{\alpha}}(\mathbf{x}_n)\right\}\right) \qquad d_n = \frac{\exp(-y_n f_{\boldsymbol{\alpha}}(\mathbf{x}_n))}{1 + \exp(-y_n f_{\boldsymbol{\alpha}}(\mathbf{x}_n))}$$

- Least Square Boosting (Friedman, 1999)

$$\mathrm{G}[f_{\boldsymbol{\alpha}}] = \sum_{n=1}^{N} (y_n - f_{\boldsymbol{\alpha}}(\mathbf{x}_n))^2 \qquad d_n = y_n - f_{\boldsymbol{\alpha}}(\mathbf{x}_n)$$

- Boosting with Soft Margins (Rätsch et al., 2001a, 2001b)

$$\mathrm{G}[f_{\boldsymbol{\alpha}}] = \sum_{n=1}^{N} \log\left(1 + \exp\left\{1 - y_n f_{\boldsymbol{\alpha}}(\mathbf{x}_n)\right\}\right) \qquad d_n = \frac{\exp(1 - y_n f_{\boldsymbol{\alpha}}(\mathbf{x}_n))}{1 + \exp(1 - y_n f_{\boldsymbol{\alpha}}(\mathbf{x}_n))}$$

Coordinate Descent

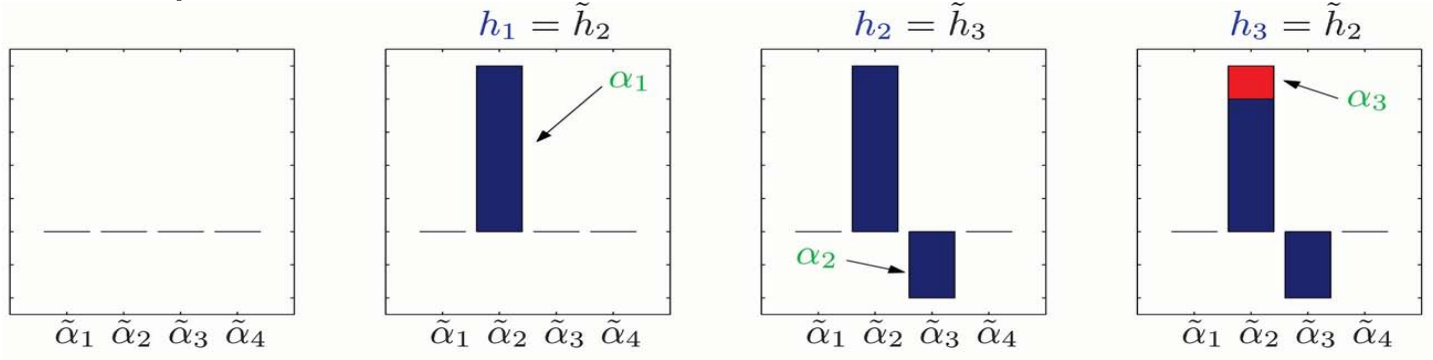
- Task: minimize a loss function of the form:

$$\mathrm{F}(\tilde{\boldsymbol{\alpha}}) = \mathrm{F}(\tilde{\alpha}_1, \ldots, \tilde{\alpha}_J)$$

- Particularly simple strategy: Iteratively select a coordinate, say the j-th, and find $\tilde{\alpha}_j$, such that $\mathrm{F}(\tilde{\alpha}_1, \ldots, \tilde{\alpha}_j, \ldots, \tilde{\alpha}_J)$ is minimized.
- How to choose j? Different strategies:
  - At random.
  - Round robin fashion.
  - Gauss-Southwell Scheme (GS): select component with largest, absolute gradient value.
  - Maximum-Improvement Scheme (MI): Choose the coordinate such, that the objective decreases maximally.
- Converges under certain conditions

Leveraging as Coordinate Descent

- Leveraging implements a coordinate gradient descent method in $\tilde{\alpha}$-coefficient space $\mathbf{R}^J$ to minimize $\mathrm{G}[f_{\tilde{\alpha}}]$
- Example: four different hypotheses, three Leveraging iterations



$\tilde{h}_2$ is selected twice and its weight is corrected in the 3rd iteration.

- If the learner selects a hypothesis more than once, we merge the weights by:

$$\tilde{\alpha}_j^{(t)} = \sum_{r=1}^{t} \alpha_r \mathrm{I}(h_r = \tilde{h}_j), \quad j = 1, \ldots, J, \qquad (1)$$

Base Learner and Selection Schemes

- If the base learner minimizes the $\mathbf{d}^{(t)}$-weighted training error ϵ_t, then it
  - maximizes the edge γ_t (assume $\tilde{h}_j(\mathbf{x}) \in \{-1, +1\}$) and
  - finds the hypothesis $h \in \mathcal{H}$ that has the largest gradient component (Gauss-Southwell Scheme), since

$$\frac{\partial \, \mathrm{G}[f_{\tilde{\boldsymbol{\alpha}}^{(t-1)}}]}{\partial \tilde{\alpha}_j} = \sum_{n=1}^{N} d_n^{(t)} \tilde{h}_j(\mathbf{x}_n) = \gamma_t,$$

- The base learner can be used as an "Oracle" to select new hypotheses/directions for coordinate descent.
- In confidence rated boosting (Schapire & Singer, 98) one uses the base learner to find the hypothesis that minimizes G (MI Scheme).

Preliminaries & Conditions

- The general form we consider is:

$$\min_{\tilde{\boldsymbol{\alpha}} \in \mathcal{S}} \mathrm{F}(\tilde{\boldsymbol{\alpha}}), \quad \text{where} \quad \mathrm{F}(\tilde{\boldsymbol{\alpha}}) = \mathrm{G}(H\tilde{\boldsymbol{\alpha}}) + \boldsymbol{\kappa}^\top \tilde{\boldsymbol{\alpha}}, \qquad (2)$$

  where G twice continously differenctiable and strictly convex on $\mathrm{dom}\ \mathrm{G}$, H has no zero columns, and $\mathcal{S}$ is a (not necessarily bounded) box constrained set.

- When/how will coordinate descent for such an loss function (think: AdaBoost) converge? **Only depends on base learner!**

- AdaBoost, Logistic Regression, LS-Boost all have such loss functions where $\boldsymbol{\kappa}$ is zero (hence: forget $\boldsymbol{\kappa}$ for now).

- Here we will only consider finite hypothesis sets (closed under complementation).

Condition: δ-Optimality

Let $\mathcal{H}$ be a finite, complementation closed hypothesis set and G be a functional as e.g. in AdaBoost.

A weak learner is called δ-optimal, if it always returns a hypothesis $\tilde{h}_j \in \mathcal{H}$ that satisfies

1. $\gamma_j \geq \delta \max_{j'=1,\ldots,J} \gamma_{j'}$, where $\gamma_j = \sum_{n=1}^{N} d_n \tilde{h}_j(\mathbf{x}_n)$,

or

2. $\mathrm{G}(\tilde{\boldsymbol{\alpha}}^t) - \mathrm{G}(\tilde{\boldsymbol{\alpha}}^{t+1,j}) \geq \delta \max_{j'=1,\ldots,J} \left(\mathrm{G}(\tilde{\boldsymbol{\alpha}}^t) - \mathrm{G}(\tilde{\boldsymbol{\alpha}}^{t+1,j'}) \right)$

for some fixed $\delta \in (0, 1]$ and all $\mathbf{d} = \nabla \mathrm{G}(\boldsymbol{\alpha})$.

$\Rightarrow$ Approximative Gauss-Southwell vs. Maximum Improvement Scheme

Convergence Theorem for Leveraging

Theorem: Given a training sample S, a finite, complementation closed hypothesis set $\mathcal{H}$. Let $\mathrm{G} : \mathbf{R}^N \to \mathbf{R}$ be twice differentiable, strictly convex and its Hessian be uniformly upper bounded. Suppose Leveraging generates a sequence of combined hypotheses $f_{\tilde{\boldsymbol{\alpha}}^0}, f_{\tilde{\boldsymbol{\alpha}}^1}, \ldots$ using a δ-optimal learning algorithm $\mathcal{L}$ for $\mathcal{H}$.

Then any limit point of $\{\tilde{\boldsymbol{\alpha}}^t\}$ is a solution and the sequence converges *linearly*.

Note: Linear convergence means, that the objective function decreases exponentially fast towards its minimum:

$$\mathrm{G}[f_{\tilde{\boldsymbol{\alpha}}^{(t)}}] - \mathrm{G}[f^*] \leq c^{-t} \quad \text{for some } c < 1$$

And what about AdaBoost?

- **Corollary** Suppose AdaBoost, Logistic Regression or LS-Boost generates a sequence of combined hypotheses $f_{\tilde{\boldsymbol{\alpha}}(1)}, f_{\tilde{\boldsymbol{\alpha}}(2)}, \ldots$ using a δ-optimal base learner. Assume solutions of the optimization problem are finite. Then $\{f_{\tilde{\boldsymbol{\alpha}}(t)}\}$ converges linearly to a solution!
- Already known for AdaBoost and Logistic Regression if the data is separable (PAC boosting property). But here it also holds for nonseparable data and a large class of loss functions!