Boosting, Margins and Beyond

Gunnar Rätsch

Friedrich Miescher Laboratory, Max Planck Society,
Tübingen, Germany

May 20, 2008

17^{th} Annual Belgian-Dutch Conference on Machine Learning
Benelearn 2008, Spa, Belgium

The AdaBoost Algorithm

Friedrich Miescher Laboratory
of the Max Planck Society

- Maintain a distribution $\mathbf{d}^t = [d_1^t, \ldots, d_N^t]$ on the examples
- At iteration $t = 1, \ldots, T$:
  1. Receive a "weak" hypothesis h_t
  2. Update $\mathbf{d}^t$ to $\mathbf{d}^{t+1}$, put more weights on "hard" examples
- Output a convex combination of the weak hypotheses

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

[Freund & Schapire, 1997]

The AdaBoost Algorithm

- Maintain a distribution $\mathbf{d}^t = [d_1^t, \ldots, d_N^t]$ on the examples
- At iteration $t = 1, \ldots, T$:
  1. Receive a "weak" hypothesis h_t
  2. Update $\mathbf{d}^t$ to $\mathbf{d}^{t+1}$, put more weights on "hard" examples
- Output a convex combination of the weak hypotheses

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

[Freund & Schapire, 1997]

The AdaBoost Algorithm

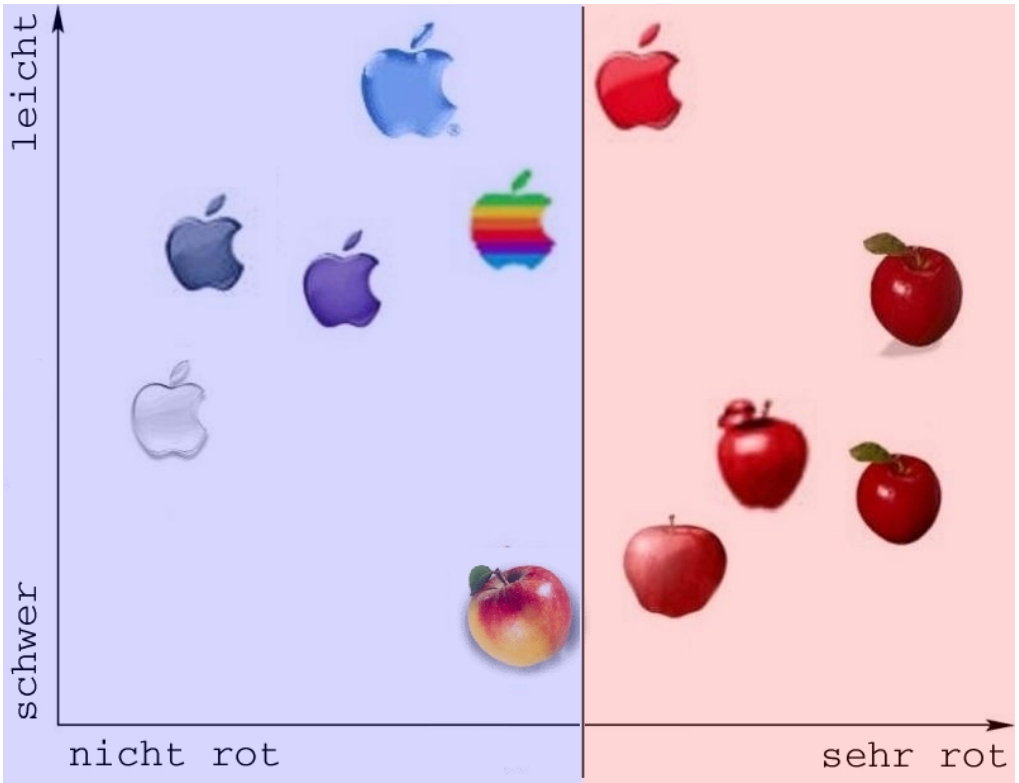
- Maintain a distribution $\mathbf{d}^t = [d_1^t, \ldots, d_N^t]$ on the examples
- At iteration $t = 1, \ldots, T$:
  1. Receive a "weak" hypothesis h_t
  2. Update $\mathbf{d}^t$ to $\mathbf{d}^{t+1}$, put more weights on "hard" examples
- Output a convex combination of the weak hypotheses

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

[Freund & Schapire, 1997]

Boosting: 1st Iteration

Friedrich Miescher Laboratory
of the Max Planck Society



First hypothesis:

- Error rate: $\frac{2}{11}$

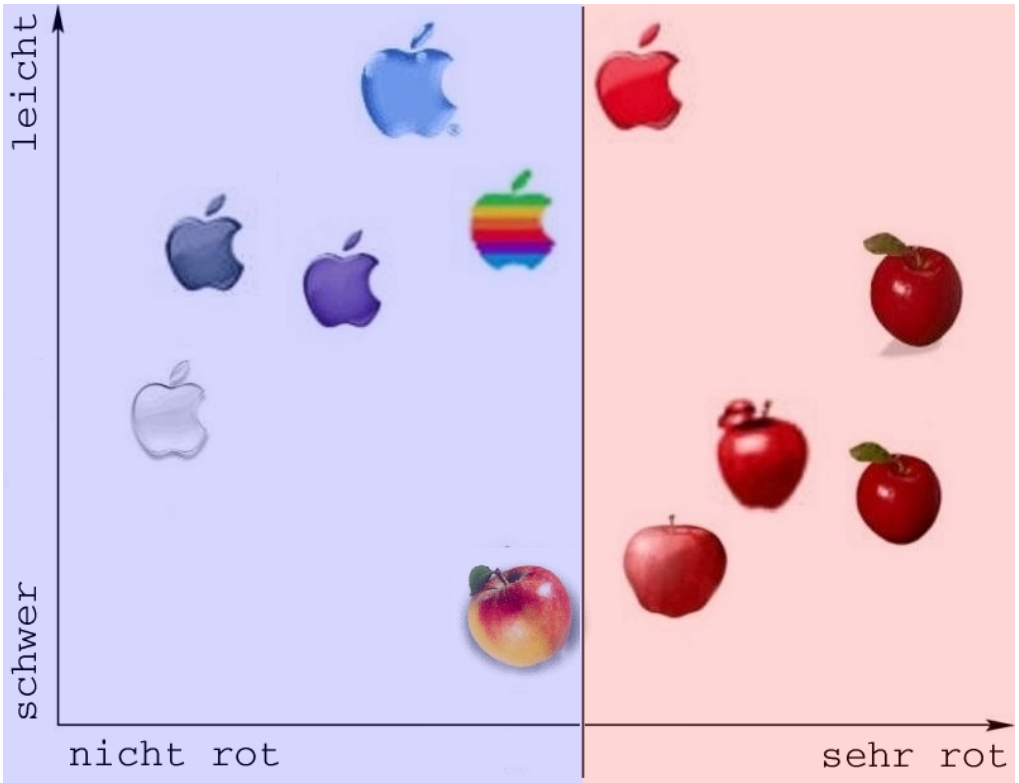
$$\epsilon_t = \sum_{n=1}^{N} d_n^t \mathbf{I}(h_t(\mathbf{x}_n) = y_n)$$

- Edge: $\frac{7}{11}$

$$\gamma_t = \sum_{n=1}^{N} d_n^t y_n h_t(\mathbf{x}_n) = 1 - 2\epsilon_t$$

Boosting: 1st Iteration

Friedrich Miescher Laboratory
of the Max Planck Society



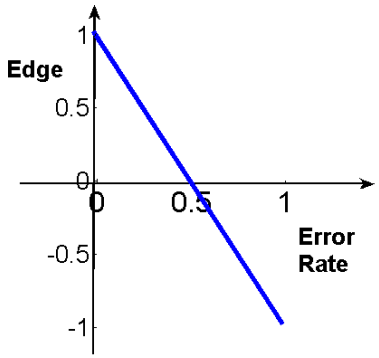
First hypothesis:

- Error rate: $\frac{2}{11}$

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

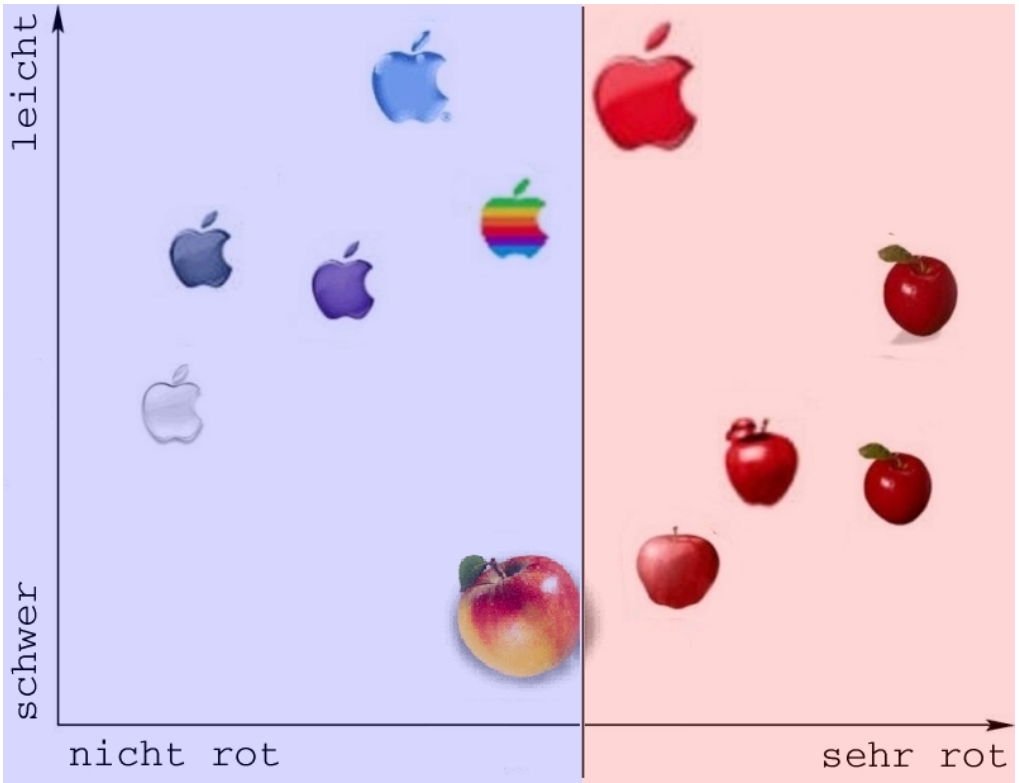
- Edge: $\frac{7}{11}$

$$\gamma_t = \sum_{n=1}^{N} d_n^t y_n h_t(\mathbf{x}_n)$$
$$= 1 - 2\epsilon_t$$



Update Distribution

Friedrich Miescher Laboratory
of the Max Planck Society



Misclassified examples
$\Rightarrow$ Increase weights

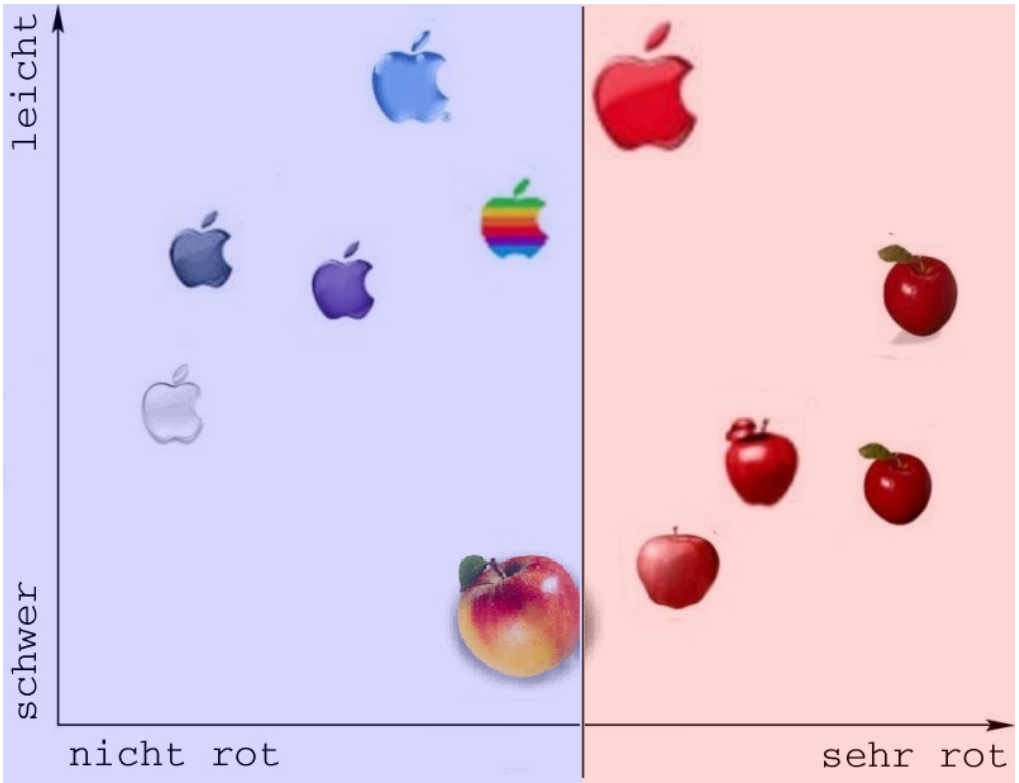
How?

After update:

- Error rate:
  $\epsilon(h_t, \mathbf{d}^{t+1}) = \frac{1}{2}$
- Edge:
  $\gamma(h_t, \mathbf{d}^{t+1}) = 0$

Update Distribution

Friedrich Miescher Laboratory
of the Max Planck Society



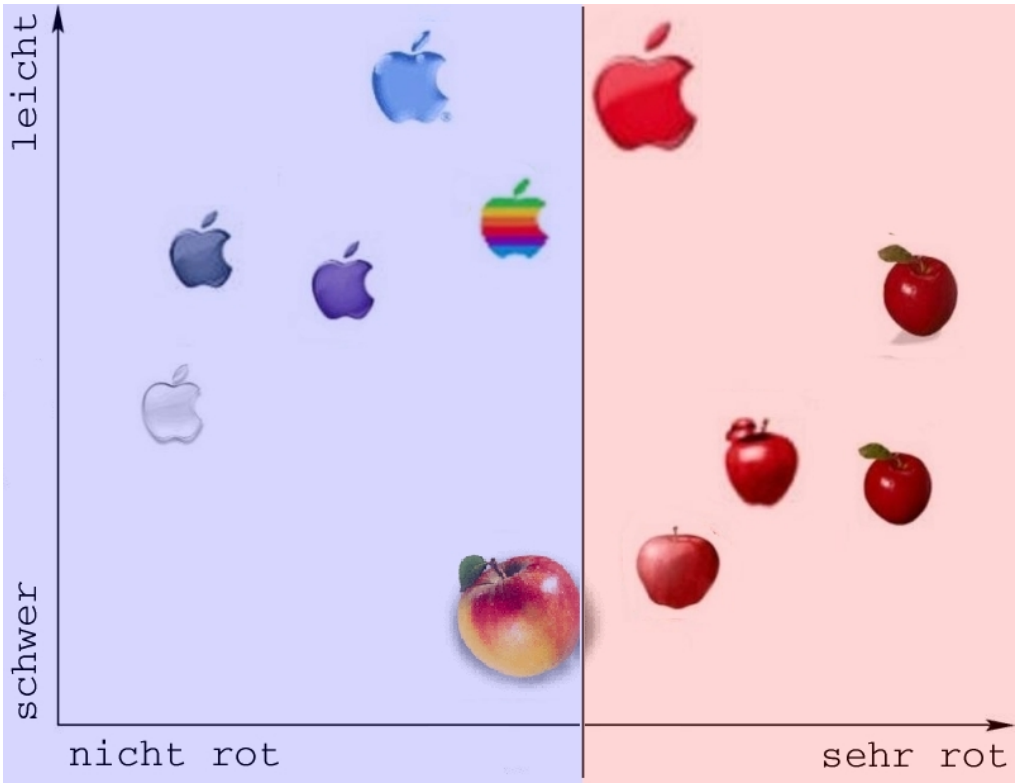
Misclassified examples
$\Rightarrow$ Increase weights

How?

After update:

- Error rate:
  $\epsilon(h_t, \mathbf{d}^{t+1}) = \frac{1}{2}$
- Edge:
  $\gamma(h_t, \mathbf{d}^{t+1}) = 0$

Update Distribution



Misclassified examples
$\Rightarrow$ Increase weights

How?

After update:

- Error rate:
  $\epsilon(h_t, \mathbf{d}^{t+1}) = \frac{1}{2}$
- Edge:
  $\gamma(h_t, \mathbf{d}^{t+1}) = 0$

Boosting as Entropy Projection

Minimize relative entropy to last distribution subject to constraints

$$\begin{aligned} \min_{\mathbf{d}} \quad & \Delta(\mathbf{d}, \mathbf{d}^t) \\ \text{s.t.} \quad & \sum_{n=1}^{N} d_n y_n h_t(\mathbf{x}_n) = 0 \\ & \mathbf{d} \in \mathcal{P}^N, \end{aligned}$$

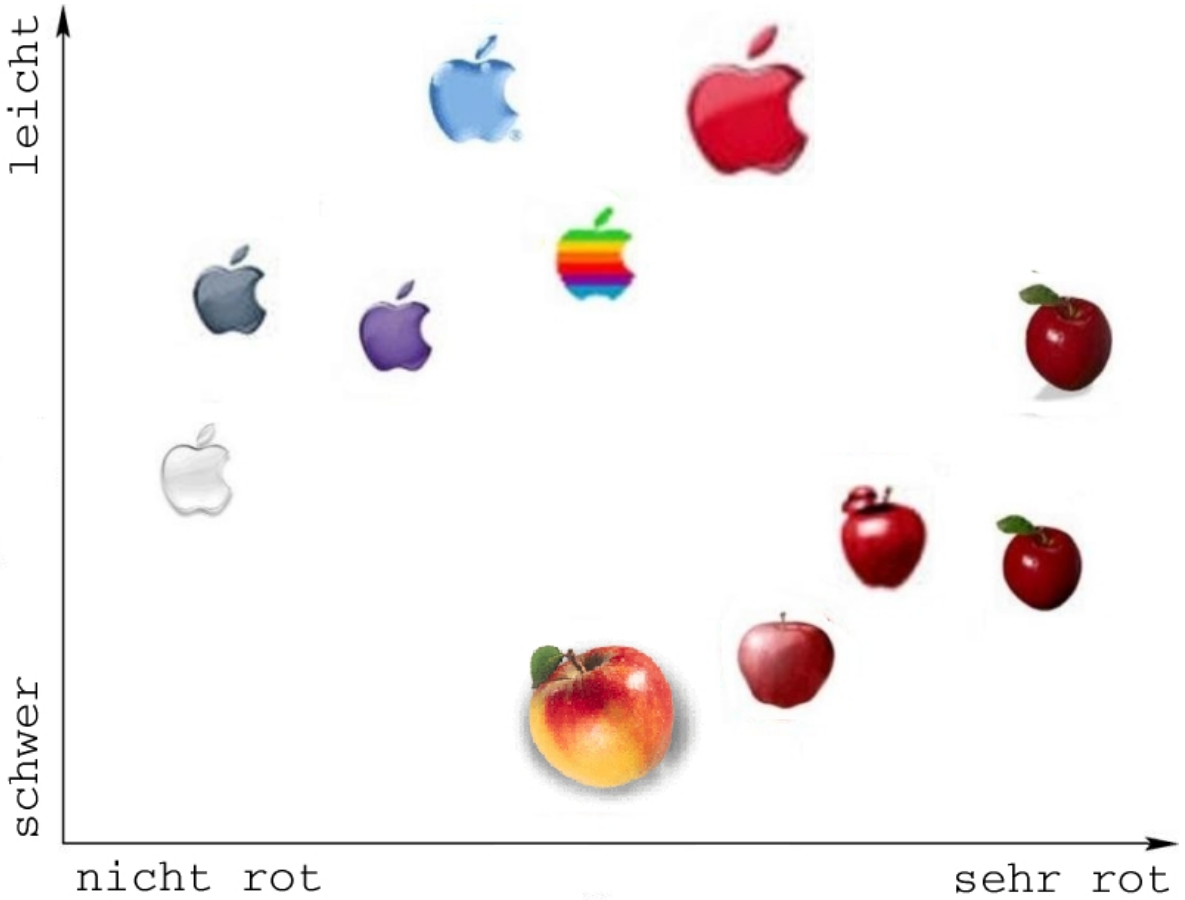
where

- $\Delta(\mathbf{d}, \mathbf{d}^t) = \sum_{n=1}^{N} d_n \ln \frac{d_n}{d_n^t}$ and
- $\mathcal{P}^N$ is the N dimensional probability simplex.

[Lafferty, 1999; Kivinen & Warmuth, 1999]

Before 2nd Iteration

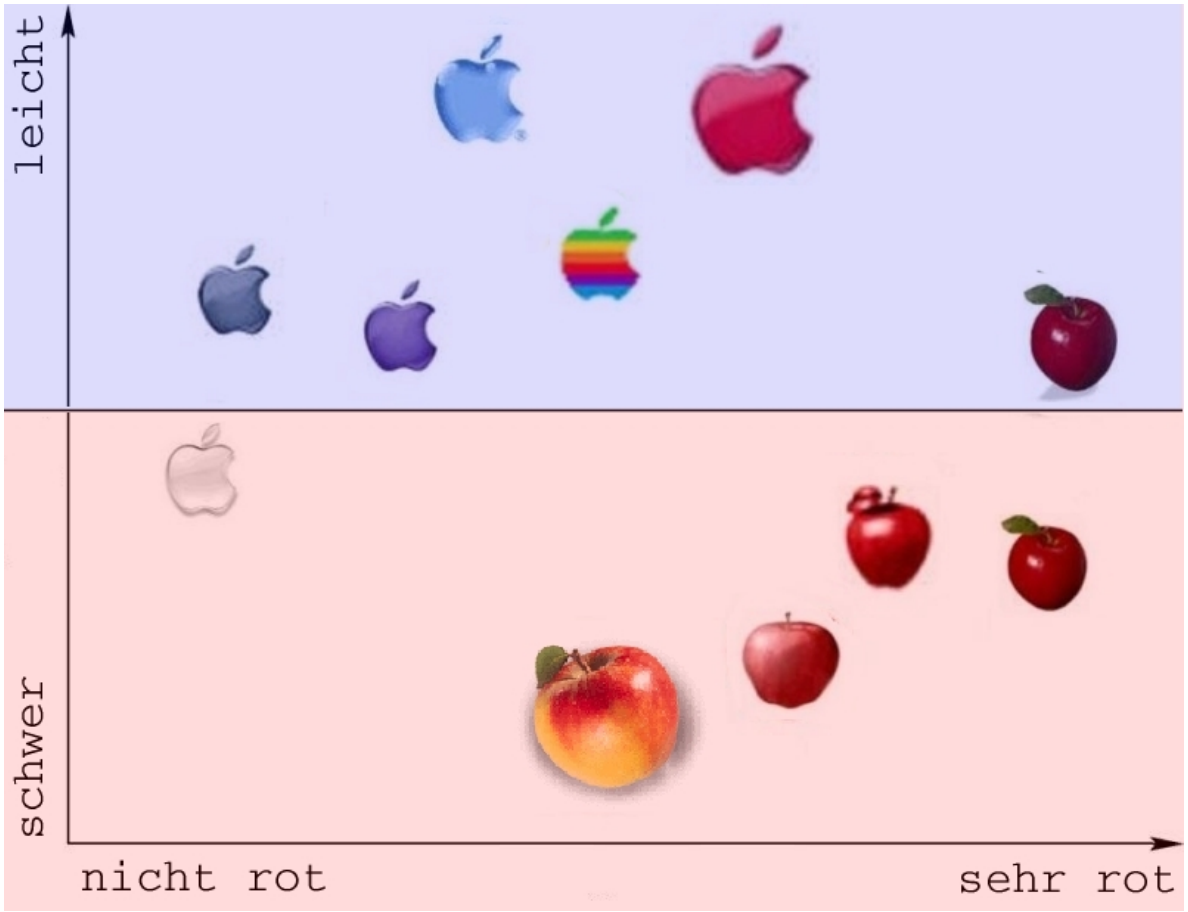
Friedrich Miescher Laboratory
of the Max Planck Society



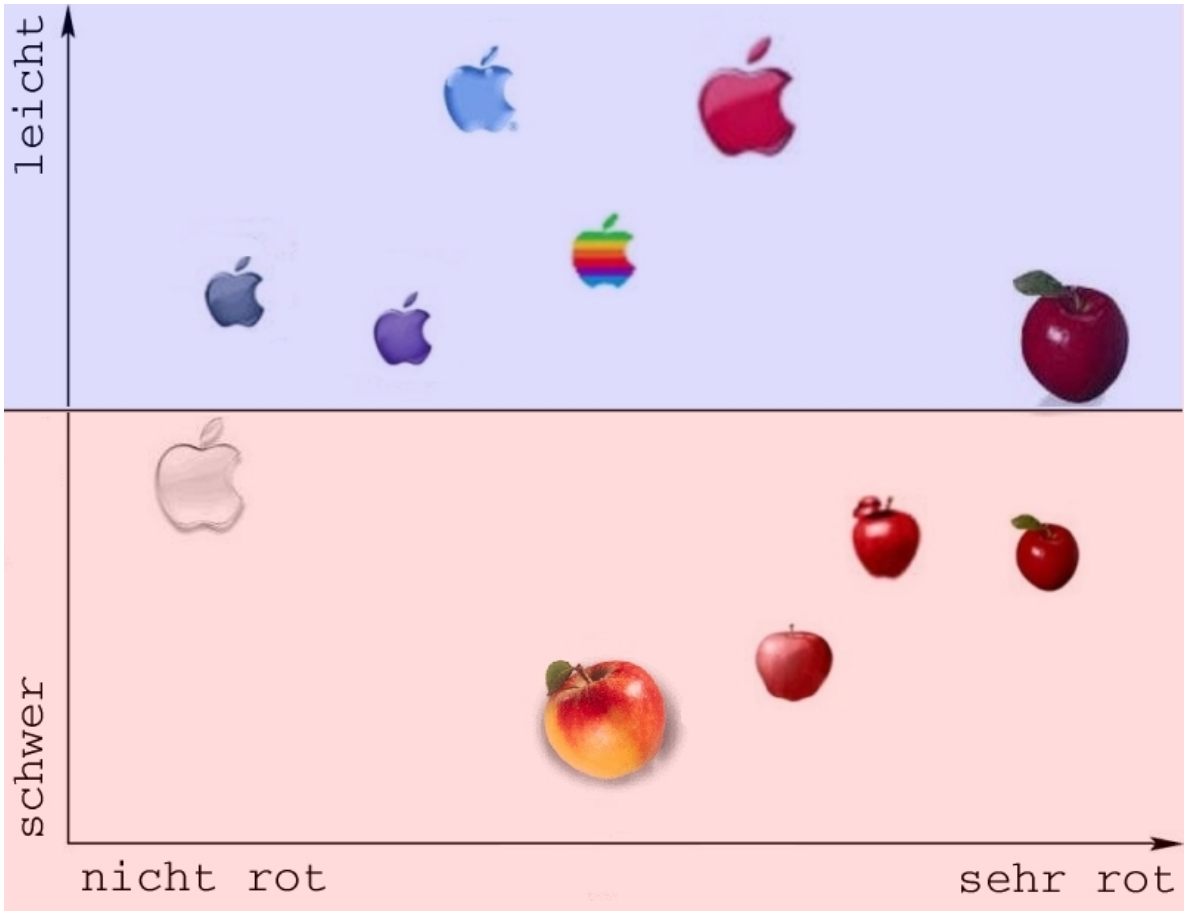
Boosting: 2nd Hypothesis

Friedrich Miescher Laboratory
of the Max Planck Society

Edge $\gamma > \Gamma$



Update Distribution



Edge $\gamma = 0$

AdaBoost update sets edge of last hypothesis to 0

Which constraints?

Friedrich Miescher Laboratory
of the Max Planck Society

Corrective Algorithm: Consider single constraint

$$\min_{\mathbf{d}\in\mathcal{P}^N} \quad \Delta(\mathbf{d}, \mathbf{d}^t)$$

$$\text{s.t.} \quad \sum_{n=1}^{N} d_n y_n h_t(\mathbf{x}_n) = 0 \qquad [\text{or} \leq 0]$$

Edge of the last hypothesis zero, i.e. $\sum_{n=1}^{N} d_n y_n h_{\mathbf{t}}(\mathbf{x}_n) = 0.$

But not of the previous iterations, e.g. $\sum_{n=1}^{N} d_n y_n h_{\mathbf{t-1}}(\mathbf{x}_n) \neq 0$

[Lafferty, 1999; Kiwinen & Warmuth, 1999]

Which constraints?

Friedrich Miescher Laboratory
of the Max Planck Society

Corrective Algorithm: Consider single constraint

$$\min_{\mathbf{d}\in\mathcal{P}^N} \quad \Delta(\mathbf{d}, \mathbf{d}^t)$$

$$\text{s.t.} \quad \sum_{n=1}^{N} d_n y_n h_t(\mathbf{x}_n) = 0 \qquad \text{for } t = 1, \ldots, T$$

Totally Corrective Algorithm: Consider all previous constraints

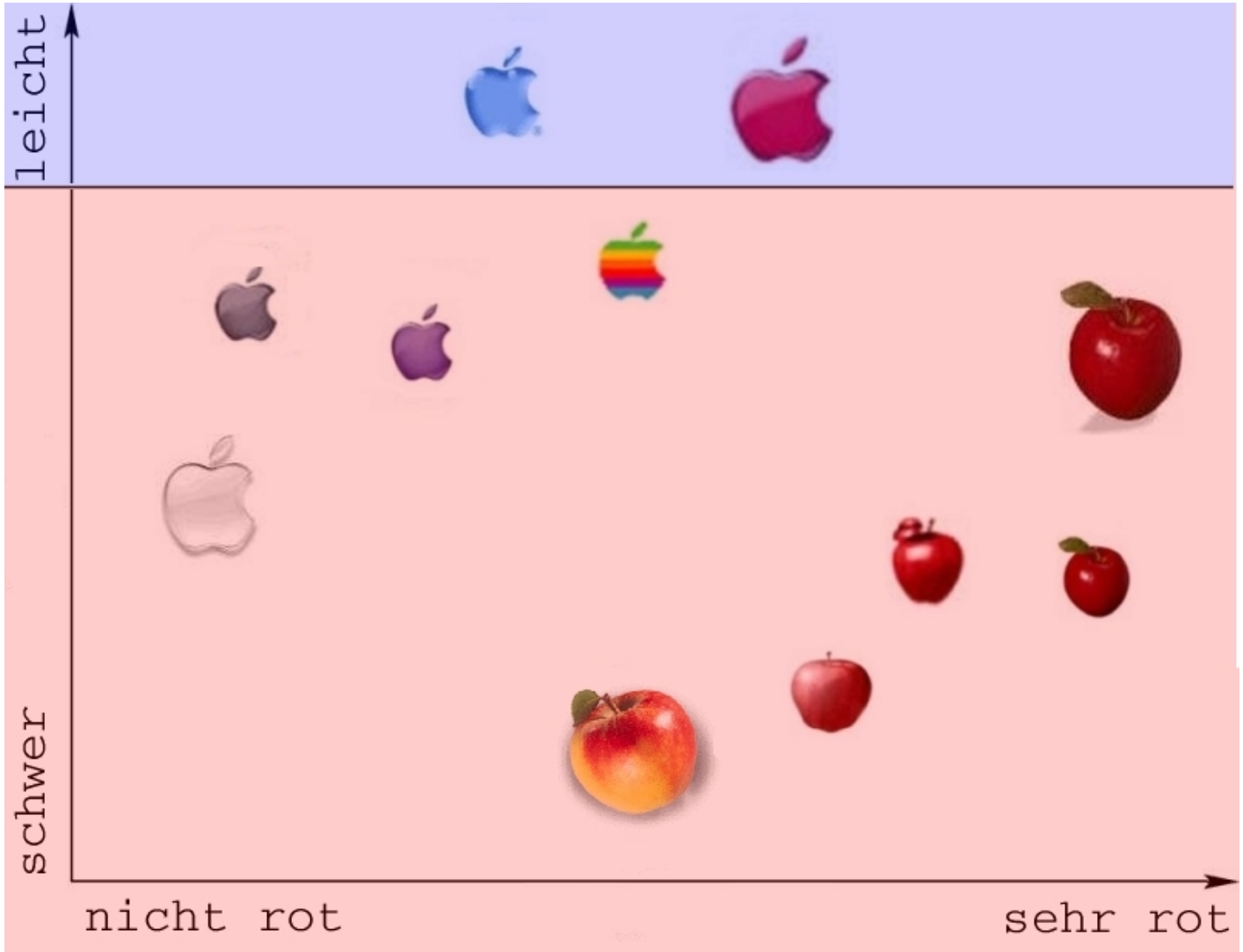
$$\min_{\mathbf{d}\in\mathcal{P}^N} \quad \Delta(\mathbf{d}, \mathbf{d}^1)$$

$$\text{s.t.} \quad \sum_{n=1}^{N} d_n y_n h_{\mathbf{r}}(\mathbf{x}_n) = 0 \qquad \text{for } \mathbf{r} = \mathbf{1}, \ldots, \mathbf{t}$$

[Lafferty, 1999; Kiwinen & Warmuth, 1999]

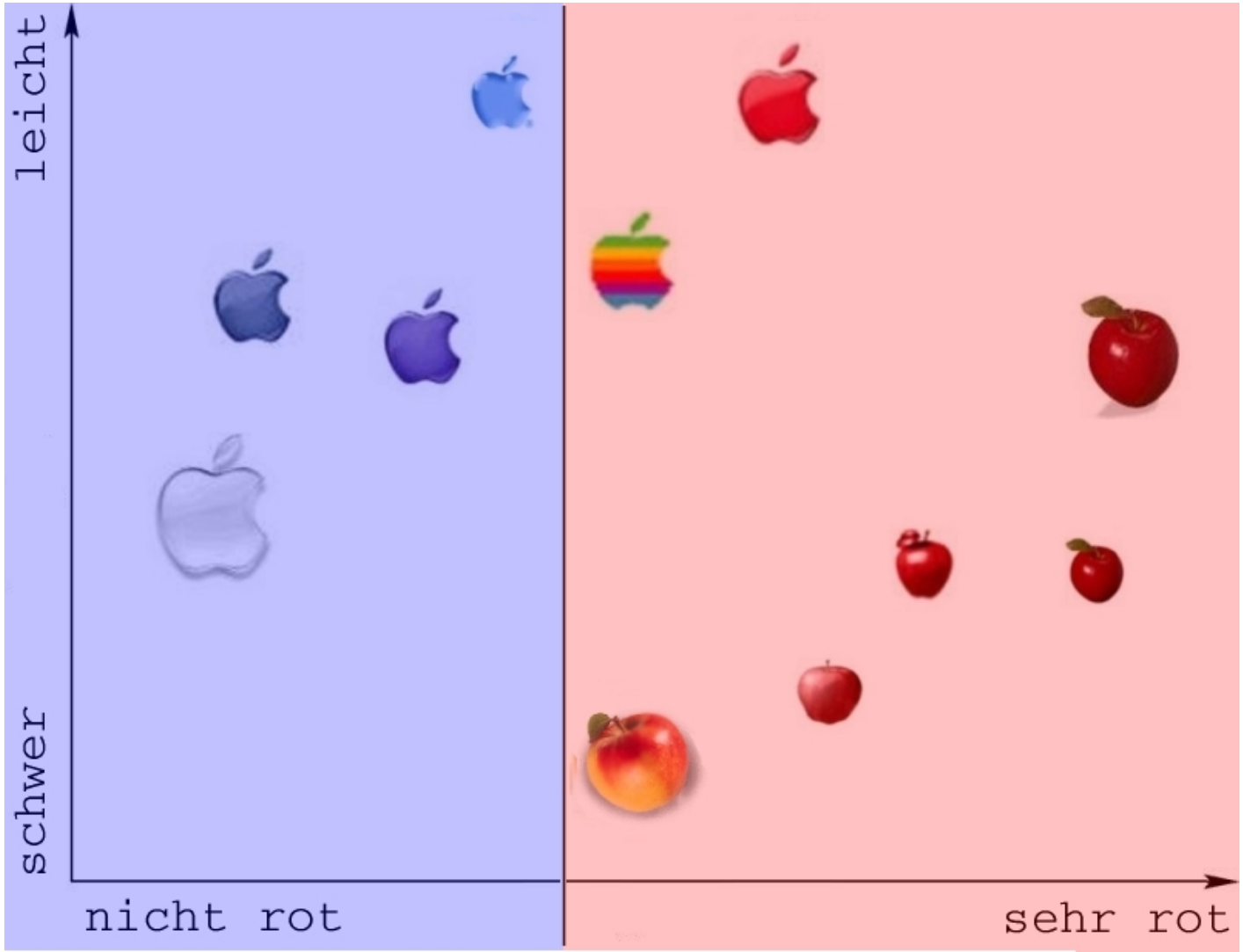
Boosting: 3nd Hypothesis

Friedrich Miescher Laboratory
of the Max Planck Society



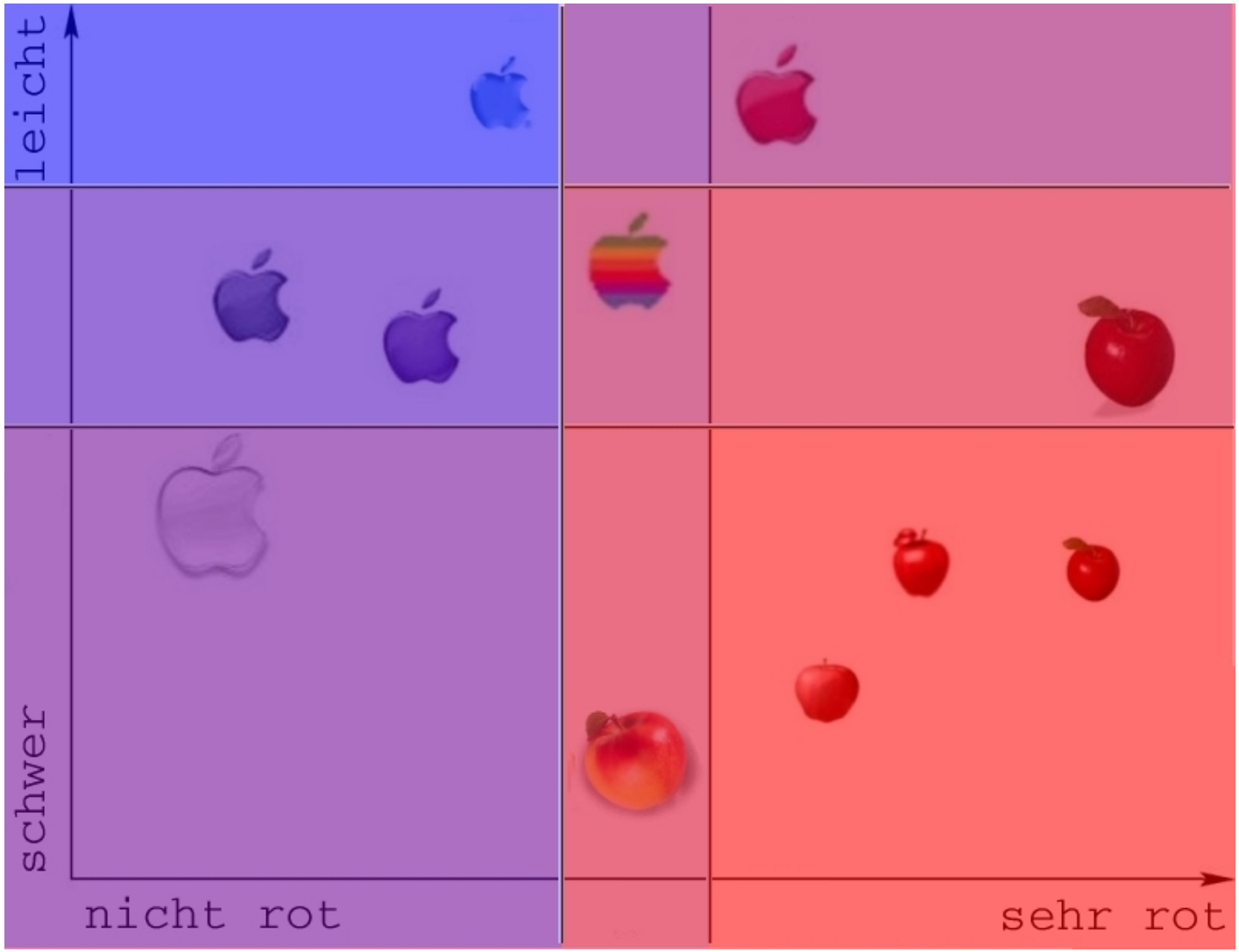
Boosting: 4th Hypothesis

Friedrich Miescher Laboratory
of the Max Planck Society



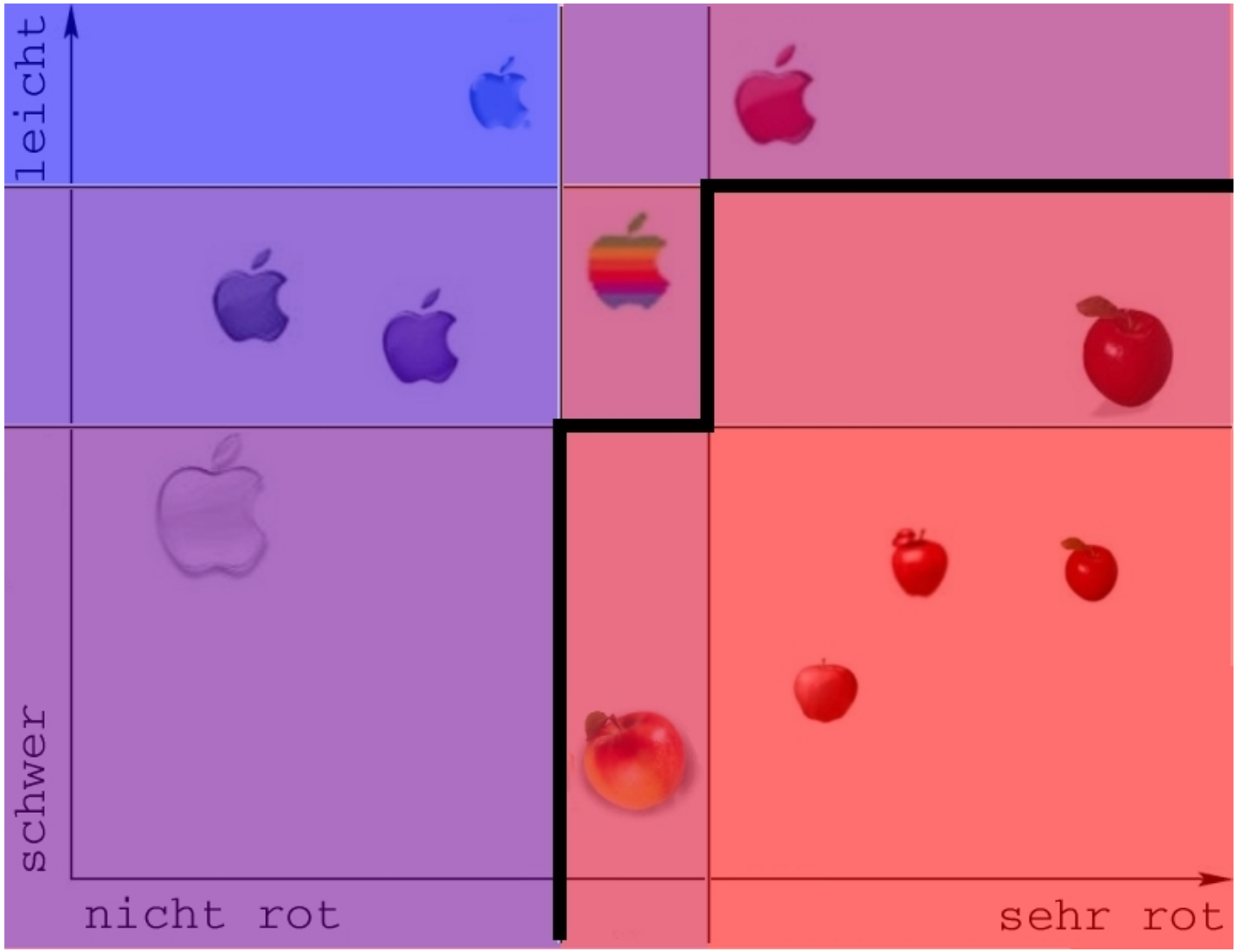
All Hypotheses

Friedrich Miescher Laboratory
of the Max Planck Society



Decision: $f_\alpha(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x}) > 0$?

Friedrich Miescher Laboratory
of the Max Planck Society



AdaBoost

1. **Input:** $S = \langle (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N) \rangle$
2. **Initialize:** $d_n^1 = 1/N$ for all $n = 1 \ldots N$
3. **Do for** $t = 1, \ldots$
   1. Train classifier on $\{S, \mathbf{d}^t\}$ and obtain hypothesis $h_t : \mathbf{x} \mapsto \{\pm 1\}$ and let $u_i^t = y_i h_t(\mathbf{x}_i)$
   2. Calculate the edge γ_t of h_t: $\gamma_\mathbf{t} = \mathbf{d^t} \cdot \mathbf{u^t}$
   3. Compute α_t: $\alpha_t = \frac{1}{2} \log \frac{1+\gamma_t}{1-\gamma_t}$
   4. $\mathbf{d}^{t+1} = \mathbf{d}^t \exp\left(-\alpha_t y_i h_y(\mathbf{x}_i)\right) / Z$
   5. **If** $\alpha_t = \infty$ then $T = t$ and break
4. **Output:** $f_\alpha(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})$.

AdaBoost

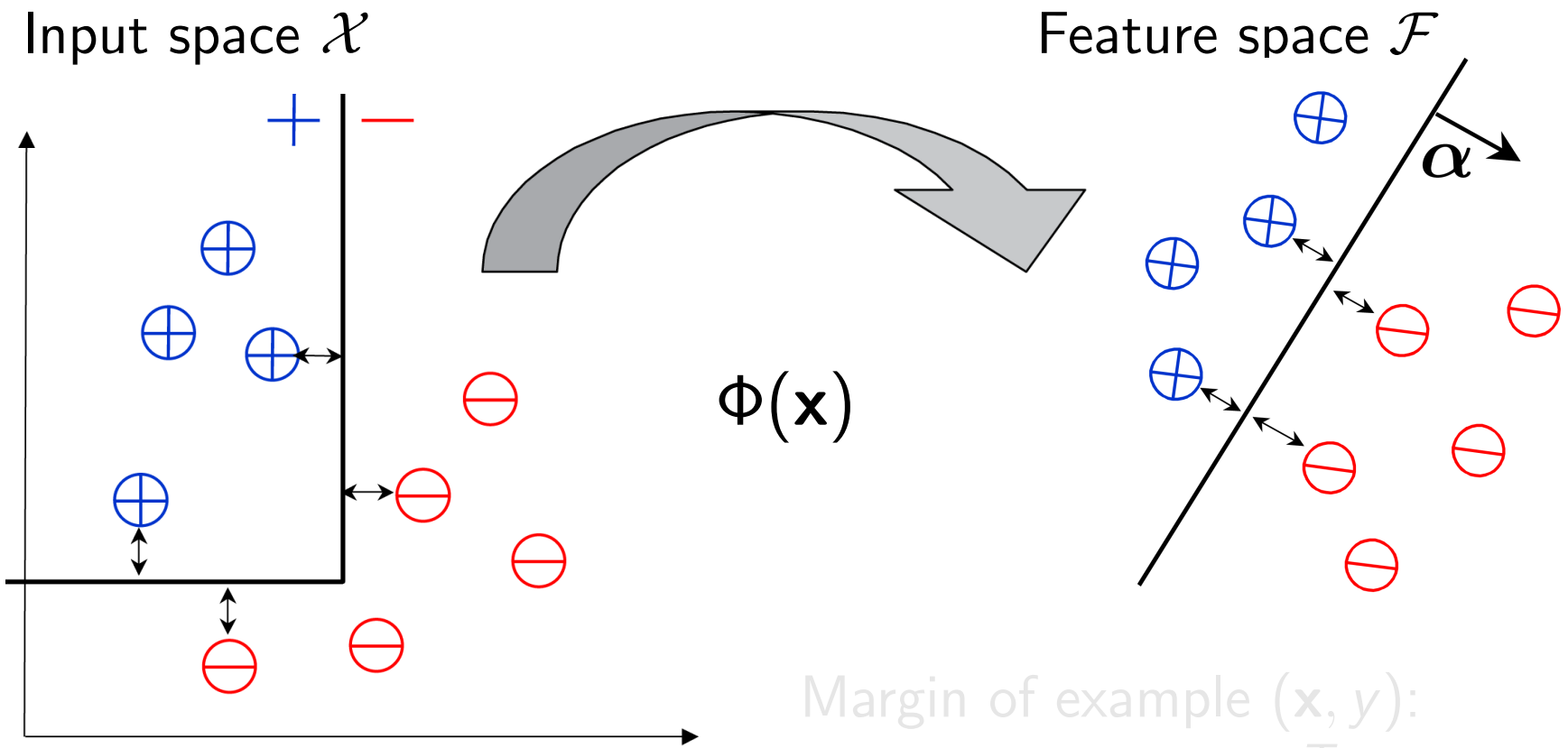
1. **Input:** $S = \langle (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N) \rangle$
2. **Initialize:** $d_n^1 = 1/N$ for all $n = 1 \ldots N$
3. **Do for** $t = 1, \ldots$
   1. Train classifier on $\{S, \mathbf{d}^t\}$ and obtain hypothesis $h_t : \mathbf{x} \mapsto [-1, 1]$ and let $u_i^t = y_i h_t(\mathbf{x}_i)$
   2. Calculate the edge γ_t of h_t: $\gamma_\mathbf{t} = \mathbf{d^t} \cdot \mathbf{u^t}$
   3. **Optimization Problem**

$$\begin{aligned} \mathbf{d}^{t+1} &= \operatorname{argmin} \Delta(\mathbf{d}, \mathbf{d}^t) \\ \text{with} \quad & \mathbf{d} \in \{\mathbf{d} \in \mathcal{P}^N \mid \mathbf{d} \cdot \mathbf{u}^t = 0\}. \end{aligned}$$

4. **Output:** $f_\alpha(\mathbf{x}) = \sum_{t=1}^T \alpha_t h_t(\mathbf{x})$.

Linear Separation and Large Margins

Friedrich Miescher Laboratory
of the Max Planck Society



Linear separation in $\mathcal{F}$ is
nonlinear separation in $\mathcal{X}$

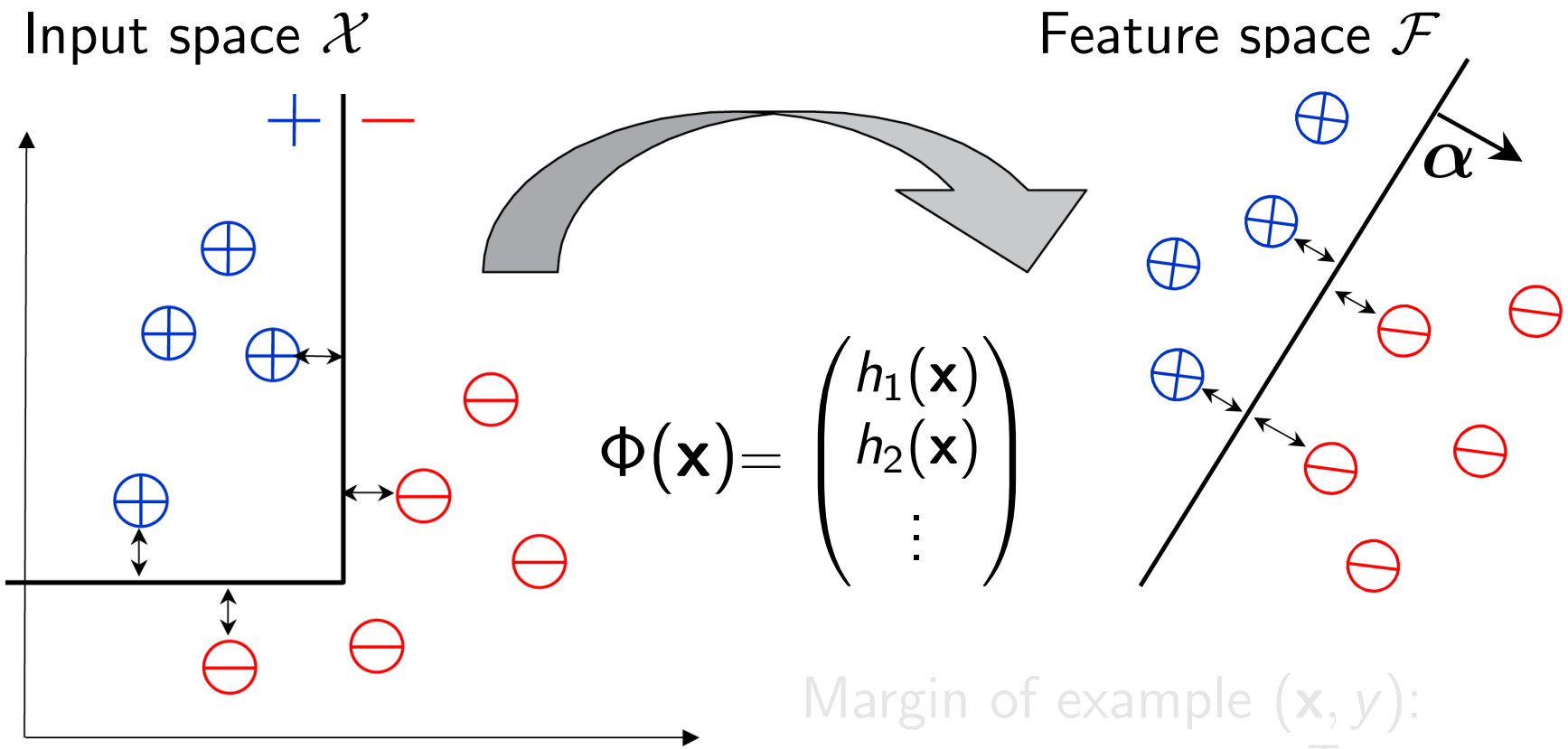
Margin of example $(\mathbf{x}, y)$:

$$y\langle \Phi(\mathbf{x}), \alpha \rangle = y \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})$$

[Schapire et al., 1998, Breiman, 1999, Mangasarian, 1999; G.R., Mika, Schölkopf & Müller, 2002]

Linear Separation and Large Margins

Friedrich Miescher Laboratory
of the Max Planck Society



Linear separation in $\mathcal{F}$ is
nonlinear separation in $\mathcal{X}$

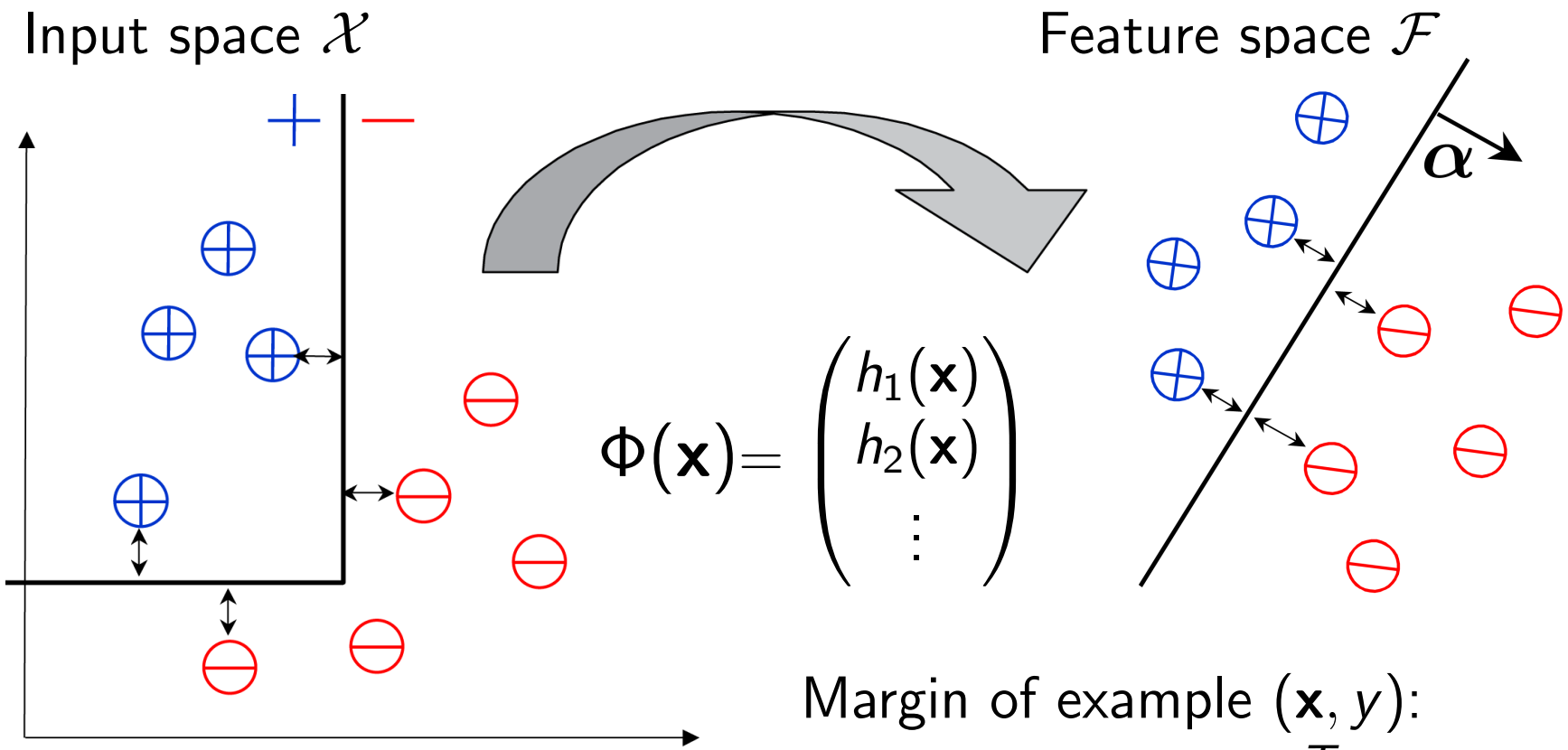
Margin of example $(\mathbf{x}, y)$:

$$y\langle \Phi(\mathbf{x}), \alpha \rangle = y \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})$$

[Schapire et al., 1998, Breiman, 1999, Mangasarian, 1999; G.R., Mika, Schölkopf & Müller, 2002]

Linear Separation and Large Margins

Friedrich Miescher Laboratory
of the Max Planck Society



Linear separation in $\mathcal{F}$ is
nonlinear separation in $\mathcal{X}$

Margin of example $(\mathbf{x}, y)$:

$$y\langle \Phi(\mathbf{x}), \boldsymbol{\alpha} \rangle = y \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})$$

[Schapire et al., 1998, Breiman, 1999, Mangasarian, 1999; G.R., Mika, Schölkopf & Müller, 2002]

Margin vs. Edge

Margin

- Measure for "confidence" in prediction of example n: $y_n f_\alpha(\mathbf{x}_n)$
- Depends on hypothesis weighting α

Edge

- "Goodness" of hypothesis returned by learner: $\gamma = 1 - 2\epsilon$
- Depends on distribution $\mathbf{d}$ on examples

What is the connection?

von Neumann's Minimax-Theorem

$$\textbf{minimum edge} \quad = \quad \textbf{maximum margin}$$

[von Neumann, 1928, Breiman, 1999]

Margin vs. Edge

Friedrich Miescher Laboratory
of the Max Planck Society

Margin

- Measure for "confidence" in prediction of example n: $y_n f_{\boldsymbol{\alpha}}(\mathbf{x}_n)$
- Depends on hypothesis weighting $\boldsymbol{\alpha}$

Edge

- "Goodness" of hypothesis returned by learner: $\gamma = 1 - 2\epsilon$
- Depends on distribution $\mathbf{d}$ on examples

What is the connection?

von Neumann's Minimax-Theorem

minimum edge = maximum margin

[von Neumann, 1928, Breiman, 1999]

Margin vs. Edge

Friedrich Miescher Laboratory
of the Max Planck Society

Margin

- Measure for "confidence" in prediction of example n: $y_n f_\alpha(\mathbf{x}_n)$
- Depends on hypothesis weighting α

Edge

- "Goodness" of hypothesis returned by learner: $\gamma = 1 - 2\epsilon$
- Depends on distribution $\mathbf{d}$ on examples

What is the connection?

von Neumann's Minimax-Theorem

$$\text{minimum edge} \quad = \quad \text{maximum margin}$$

[von Neumann, 1928, Breiman, 1999]

Margin vs. Edge

Margin

- Measure for "confidence" in prediction of example n: $y_n f_\alpha(\mathbf{x}_n)$
- Depends on hypothesis weighting α

Edge

- "Goodness" of hypothesis returned by learner: $\gamma = 1 - 2\epsilon$
- Depends on distribution $\mathbf{d}$ on examples

What is the connection?

von Neumann's Minimax-Theorem

$$\textbf{minimum edge} \quad = \quad \textbf{maximum margin}$$

[von Neumann, 1928, Breiman, 1999]

von Neumann's Minimax-Theorem

Friedrich Miescher Laboratory
of the Max Planck Society

Set of examples $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$
and hypotheses set $\mathcal{H}^t = \{h_1, \ldots, h_t\}$,

$$\text{maximum margin: } \rho_{\mathbf{t}}^* = \max_{\boldsymbol{\alpha} \in \mathcal{P}^t} \underbrace{\min_n \overbrace{y_n f_{\boldsymbol{\alpha}}(\mathbf{x}_n)}^{\text{margin of } (\mathbf{x}_n, y_n)}}_{\text{margin of } S}$$

$$\text{minimum edge: } \gamma_{\mathbf{t}}^* = \min_{\mathbf{d} \in \mathcal{P}^N} \underbrace{\max_{h \in \mathcal{H}^t} \overbrace{\gamma(h, \mathbf{d})}^{\text{edge of } h}}_{\text{edge of } \mathcal{H}^t}$$

$$\text{Duality: } \gamma_{\mathbf{t}}^* = \rho_{\mathbf{t}}^*$$

[von Neumann, 1928]

von Neumann's Minimax-Theorem

Friedrich Miescher Laboratory
of the Max Planck Society

Set of examples $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$
and hypotheses set $\mathcal{H}^t = \{h_1, \ldots, h_t\}$,

$$\text{maximum margin: } \rho_{\mathbf{t}}^* = \max_{\boldsymbol{\alpha} \in \mathcal{P}^t} \underbrace{\min_n \overbrace{y_n f_{\boldsymbol{\alpha}}(\mathbf{x}_n)}^{\text{margin of } (\mathbf{x}_n, y_n)}}_{\text{margin of } S}$$

$$\text{minimum edge: } \gamma_{\mathbf{t}}^* = \min_{\mathbf{d} \in \mathcal{P}^N} \underbrace{\max_{h \in \mathcal{H}^t} \overbrace{\gamma(h, \mathbf{d})}^{\text{edge of } h}}_{\text{edge of } \mathcal{H}^t}$$

Duality: $\gamma_{\mathbf{t}}^* = \rho_{\mathbf{t}}^*$

[von Neumann, 1928]

von Neumann's Minimax-Theorem

Friedrich Miescher Laboratory
of the Max Planck Society

Set of examples $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$
and hypotheses set $\mathcal{H}^t = \{h_1, \ldots, h_t\}$,

$$\text{maximum margin: } \rho_{\mathbf{t}}^* = \max_{\boldsymbol{\alpha} \in \mathcal{P}^t} \underbrace{\min_n \overbrace{y_n f_{\boldsymbol{\alpha}}(\mathbf{x}_n)}^{\text{margin of } (\mathbf{x}_n, y_n)}}_{\text{margin of } S}$$

$$\text{minimum edge: } \gamma_{\mathbf{t}}^* = \min_{\mathbf{d} \in \mathcal{P}^N} \underbrace{\max_{h \in \mathcal{H}^t} \overbrace{\gamma(h, \mathbf{d})}^{\text{edge of } h}}_{\text{edge of } \mathcal{H}^t}$$

$$\text{Duality: } \gamma_{\mathbf{t}}^* = \rho_{\mathbf{t}}^*$$

[von Neumann, 1928]

How Large is the Maximal Margin?

Assumptions on Weak Learner

For any distribution $\mathbf{d}$ on the examples, the weak learner returns a hypothesis "*consistently better than random guessing*".
Hence:

- Error $\epsilon(h, \mathbf{d}) < \frac{1}{2}(1 - \rho)$ and
- Edge $\gamma(h, \mathbf{d}) > \rho$ for some $\rho > 0$.

[Breiman, 1999; Bennett et al.; G.R. et al., 2001; Rudin et al., 2005]

How Large is the Maximal Margin?

Friedrich Miescher Laboratory
of the Max Planck Society

Assumptions on Weak Learner

For any distribution $\mathbf{d}$ on the examples, the weak learner returns a hypothesis "*consistently better than random guessing*".
Hence:

- Error $\epsilon(h, \mathbf{d}) < \frac{1}{2}(1 - \Gamma)$ and
- Edge $\gamma(h, \mathbf{d}) > \Gamma$ for some $\Gamma > 0$.

Implication from Minimax Theorem

There exists $\boldsymbol{\alpha} \in \mathcal{P}^N$, such that $\rho(\boldsymbol{\alpha}) \geq \Gamma$.

Assume the weak learner does well ($\Gamma = \rho^* = \gamma^*$).
How can we maximize the margin?

[Breiman, 1999; Bennett et al.; G.R. et al., 2001; Rudin et al., 2005]

How Large is the Maximal Margin?

Friedrich Miescher Laboratory
of the Max Planck Society

Assumptions on Weak Learner

For any distribution $\mathbf{d}$ on the examples, the weak learner returns a hypothesis "*consistently better than random guessing*".
Hence:

- Error $\epsilon(h, \mathbf{d}) < \frac{1}{2}(1 - \Gamma)$ and
- Edge $\gamma(h, \mathbf{d}) > \Gamma$ for some $\Gamma > 0$.

Implication from Minimax Theorem

There exists $\boldsymbol{\alpha} \in \mathcal{P}^N$, such that $\rho(\boldsymbol{\alpha}) \geq \Gamma$.

Assume the weak learner does well ($\Gamma = \rho^* = \gamma^*$).
How can we maximize the margin?

[Breiman, 1999; Bennett et al.; G.R. et al., 2001; Rudin et al., 2005]

How Large is the Maximal Margin?

Friedrich Miescher Laboratory
of the Max Planck Society

Assumptions on Weak Learner

For any distribution $\mathbf{d}$ on the examples, the weak learner returns a hypothesis "*consistently better than random guessing*".
Hence:

- Error $\epsilon(h, \mathbf{d}) < \frac{1}{2}(1 - \gamma)$ and
- Edge $\gamma(h, \mathbf{d}) > \gamma$ for some $\gamma > 0$.

What about AdaBoost?

- Does not find maximizing $\boldsymbol{\alpha}$ (counter examples).
- But there are some guarantees, e.g.
  - $\rho(\boldsymbol{\alpha}^t) \geq 0$ after $2 \ln N / \gamma^2$ iterations [Freund & Schapire, 1997]
  - $\rho(\boldsymbol{\alpha}^t) \geq \gamma/2$ in the limit [Rätsch et al., 2001]

[Breiman, 1999; Bennett et al.; G.R. et al., 2001; Rudin et al., 2005]

How to **Maximize** the Margin?

Friedrich Miescher Laboratory
of the Max Planck Society

- **AdaBoost** does not generally maximize the margin
  - Slow and may cycle
- **Arc-GV** asymptotically maximizes the margin
  - Relatively slow, asymptotic convergence
- **LPBoost** uses a linear programming solver
  - Often very fast in practice, **no converge rates** (new: $\Omega(N)$)
- **AdaBoost*** requires $\frac{2\log(N)}{\delta^2}$ iterations to get $\rho^t \in [\rho^* - \delta, \rho^*]$
  - Slow in practice, i.e. not faster than theory predicts
- **TotalBoost** requires $\frac{2\log(N)}{\delta^2}$ iterations to get $\rho^t \in [\rho^* - \delta, \rho^*]$
  - Needs convex optimizer, fast in practice
  - **Combination of benefits**

[Breiman, 1999; Bennett et al.; G.R. et al., 2001, 2004, 2006; Rudin et al., 2004]

How to **Maximize** the Margin?

- **AdaBoost** does not generally maximize the margin
  - Slow and may cycle
- **Arc-GV** asymptotically maximizes the margin
  - Relatively slow, asymptotic convergence
- **LPBoost** uses a linear programming solver
  - Often very fast in practice, **no converge rates** (new: $\Omega(N)$)
- **AdaBoost*** requires $\frac{2\log(N)}{\delta^2}$ iterations to get $\rho^t \in [\rho^* - \delta, \rho^*]$
  - Slow in practice, i.e. not faster than theory predicts
- **TotalBoost** requires $\frac{2\log(N)}{\delta^2}$ iterations to get $\rho^t \in [\rho^* - \delta, \rho^*]$
  - Needs convex optimizer, fast in practice
  - **Combination of benefits**

[Breiman, 1999; Bennett et al.; G.R. et al., 2001, 2004, 2006; Rudin et al., 2004]

How to **Maximize** the Margin?

Friedrich Miescher Laboratory
of the Max Planck Society

- **AdaBoost** does not generally maximize the margin
  - Slow and may cycle
- **Arc-GV** asymptotically maximizes the margin
  - Relatively slow, asymptotic convergence
- **LPBoost** uses a linear programming solver
  - Often very fast in practice, **no converge rates** (new: $\Omega(N)$)
- **AdaBoost*** requires $\frac{2\log(N)}{\delta^2}$ iterations to get $\rho^t \in [\rho^* - \delta, \rho^*]$
  - Slow in practice, i.e. not faster than theory predicts
- **TotalBoost** requires $\frac{2\log(N)}{\delta^2}$ iterations to get $\rho^t \in [\rho^* - \delta, \rho^*]$
  - Needs convex optimizer, fast in practice
  - **Combination of benefits**

[Breiman, 1999; Bennett et al.; G.R. et al., 2001, 2004, 2006; Rudin et al., 2004]

How to **Maximize** the Margin?

- **AdaBoost** does not generally maximize the margin
  - Slow and may cycle
- **Arc-GV** asymptotically maximizes the margin
  - Relatively slow, asymptotic convergence
- **LPBoost** uses a linear programming solver
  - Often very fast in practice, **no converge rates** (new: $\Omega(N)$)
- **AdaBoost*** requires $\frac{2\log(N)}{\delta^2}$ iterations to get $\rho^t \in [\rho^* - \delta, \rho^*]$
  - Slow in practice, i.e. not faster than theory predicts
- **TotalBoost** requires $\frac{2\log(N)}{\delta^2}$ iterations to get $\rho^t \in [\rho^* - \delta, \rho^*]$
  - Needs convex optimizer, fast in practice
  - **Combination of benefits**

[Breiman, 1999; Bennett et al.; G.R. et al., 2001, 2004, 2006; Rudin et al., 2004]

How to **Maximize** the Margin?

Friedrich Miescher Laboratory
of the Max Planck Society

- **AdaBoost** does not generally maximize the margin
  - Slow and may cycle
- **Arc-GV** asymptotically maximizes the margin
  - Relatively slow, asymptotic convergence
- **LPBoost** uses a linear programming solver
  - Often very fast in practice, **no converge rates** (new: $\Omega(N)$)
- **AdaBoost*** requires $\frac{2\log(N)}{\delta^2}$ iterations to get $\rho^t \in [\rho^* - \delta, \rho^*]$
  - Slow in practice, i.e. not faster than theory predicts
- **TotalBoost** requires $\frac{2\log(N)}{\delta^2}$ iterations to get $\rho^t \in [\rho^* - \delta, \rho^*]$
  - Needs convex optimizer, fast in practice
  - **Combination of benefits**

[Breiman, 1999; Bennett et al.; G.R. et al., 2001, 2004, 2006; Rudin et al., 2004]

How to **Maximize** the Margin?

Friedrich Miescher Laboratory
of the Max Planck Society

- **AdaBoost** does not generally maximize the margin
  - Slow and may cycle
- **Arc-GV**
  - Rela
- **LPBoos**
  - Ofte
- **AdaBoo**
  - Slow in practice, i.e. not faster than theory predicts
- **TotalBoost** requires $\frac{2\log(N)}{\delta^2}$ iterations to get $\rho^t \in [\rho^* - \delta, \rho^*]$
  - Needs convex optimizer, fast in practice
  - **Combination of benefits**

Convergence speeds can be truly different!

- LPBoost/TotalBoost may need 10-50 instead of 10,000 iterations
- LPBoost can be much slower than TotalBoost

w: $\Omega(N)$)

$^* - \delta, \rho^*]$

[Breiman, 1999; Bennett et al.; G.R. et al., 2001, 2004, 2006; Rudin et al., 2004]

Key Ingredients for Fast Convergence

1. Consider several constraints in parallel.
   (Corrective vs. totally corrective algorithms.)
   1. TotalBoost
   2. LPBoost
2. Slightly modify the optimization problem.
   (Do not set edges to zeros.)
   1. ArcGV
   2. AdaBoost*

If done in the right way, one can prove fast convergence.

Key Ingredients for Fast Convergence

1. Consider several constraints in parallel.
   (Corrective vs. totally corrective algorithms.)
   1. TotalBoost
   2. LPBoost
2. Slightly modify the optimization problem.
   (Do not set edges to zeros.)
   1. ArcGV
   2. AdaBoost*

If done in the right way, one can prove fast convergence.

Key Ingredients for Fast Convergence

1. Consider several constraints in parallel.
   (Corrective vs. totally corrective algorithms.)
   1. TotalBoost
   2. LPBoost
2. Slightly modify the optimization problem.
   (Do not set edges to zeros.)
   1. ArcGV
   2. AdaBoost*

If done in the right way, one can prove fast convergence.

Idea 2: Totally Corrective Algorithms

$$\min_{\mathbf{d}\in\mathcal{P}^N} \quad \Delta(\mathbf{d}, \mathbf{d}^1)$$

$$\text{s.t.} \quad \sum_{n=1}^{N} d_n y_n h_r(\mathbf{x}_n) = \mathbf{0} \qquad \text{for } r = 1, \ldots, t.$$

Trouble:

- Solution does not exist after a few iterations!
- Why? AdaBoost (almost) always worked!
  $\Rightarrow$ As soon as data is separated, the margin and hence the edge is larger than zero (by Min-Max-Theorem).

[Lafferty, 1999; Kiwinen & Warmuth, 1999; G.R. et al., 2001, 2004, 2006]

Idea 2: Totally Corrective Algorithms

$$\min_{\mathbf{d}\in\mathcal{P}^N} \quad \Delta(\mathbf{d}, \mathbf{d}^1)$$

$$\text{s.t.} \quad \sum_{n=1}^{N} d_n y_n h_r(\mathbf{x}_n) = \mathbf{0} \qquad \text{for } r = 1, \ldots, t.$$

Trouble:

- Solution does not exist after a few iterations!
- Why? AdaBoost (almost) always worked!
  $\Rightarrow$ As soon as data is separated, the margin and hence the edge is larger than zero (by Min-Max-Theorem).

[Lafferty, 1999; Kiwinen & Warmuth, 1999; G.R. et al., 2001, 2004, 2006]

Idea 2: Projections to $\hat{\gamma}_t$ instead of 0

Separation

$$\min_{\mathbf{d}\in\mathcal{P}^N} \quad \Delta(\mathbf{d},\mathbf{d}^t) \qquad \text{(AdaBoost)}$$

$$\text{s.t.} \quad \sum_{n=1}^{N} d_n y_n h_t(\mathbf{x}_n) \leq \mathbf{0}$$

Large Margin Separation

$$\min_{\mathbf{d}\in\mathcal{P}^N} \quad \Delta(\mathbf{d},\mathbf{d}^t) \qquad \text{(AdaBoost}^*\text{)}$$

$$\text{s.t.} \quad \sum_{n=1}^{N} d_n y_n h_t(\mathbf{x}_n) \leq \hat{\gamma}_\mathbf{t}$$

[G.R. & Warmuth, 2004]

How to choose $\hat{\gamma}_t$?

Let $\hat{\gamma}^*$ be the smallest possible $\hat{\gamma}$ that satisfies

$$\sum_{n=1}^{N} d_n y_n h_r(\mathbf{x}_n) \leq \hat{\gamma} \qquad \text{for } r = 1, \ldots, t$$

for some $\mathbf{d}$?

Different approaches:

LPBoost:	$\hat{\gamma}_t = \hat{\gamma}^*$	(projection to minimum)
ArcGV:	$\hat{\gamma}_t \leq \hat{\gamma}^*$	(below minimum)
AdaBoost*:	$\hat{\gamma}_t \geq \hat{\gamma}^* - \delta$	("above" minimum, $\delta > 0$)

[Breiman, 1999; Bennett et al. 2001; G.R. et al., 2001, 2004, 2006]

Convergence Result

Theorem

Assume

- the base learner always returns a hypothesis with edge $\geq \Gamma$ and
- we choose $\hat{\gamma}_t = \min_{r=1,\ldots t} \gamma(h_r, \mathbf{d}^r) - \delta$,

then the algorithm stops within $\left\lceil \frac{2 \log \mathbf{N}}{\delta^2} \right\rceil$ iterations with a margin of the combined hypothesis of at least $\Gamma - \delta$.

Examples:

1. AdaBoost*
2. TotalBoost

Does not apply to:

1. ArcGV
2. LPBoost

[Breiman, 1999; Bennett et al. 2001; G.R. et al., 2001, 2004, 2006]

Convergence Result

Theorem

Assume

- the base learner always returns a hypothesis with edge $\geq \Gamma$ and
- we choose $\hat{\gamma}_t = \min_{r=1,\ldots t} \gamma(h_r, \mathbf{d}^r) - \delta$,

then the algorithm stops within $\left\lceil \frac{2 \log \mathbf{N}}{\delta^2} \right\rceil$ iterations with a margin of the combined hypothesis of at least $\Gamma - \delta$.

Examples:

1. AdaBoost*
2. TotalBoost

Does not apply to:

1. ArcGV
2. LPBoost

[Breiman, 1999; Bennett et al. 2001; G.R. et al., 2001, 2004, 2006]

Convergence Result

Friedrich Miescher Laboratory
of the Max Planck Society

Theorem

Assume

- the base learner always returns a hypothesis with edge $\geq \Gamma$ and
- we choose $\hat{\gamma}_t = \min_{r=1,\ldots t} \gamma(h_r, \mathbf{d}^r) - \delta$,

then the algorithm stops within $\left\lceil \frac{2 \log \mathbf{N}}{\delta^2} \right\rceil$ iterations with a margin of the combined hypothesis of at least $\Gamma - \delta$.

Examples:

1. AdaBoost*
2. TotalBoost

Does not apply to:

1. ArcGV
2. LPBoost

[Breiman, 1999; Bennett et al. 2001; G.R. et al., 2001, 2004, 2006]

TotalBoost$_\delta$

1. **Input:** $S = \langle (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N) \rangle$, desired accuracy δ
2. **Initialize:** $d_n^1 = 1/N$ for all $n = 1 \ldots N$
3. **Do for** $t = 1, \ldots$
   1. Train classifier on $\{S, \mathbf{d}^t\}$ and obtain hypothesis $h_t : \mathbf{x} \mapsto [-1, 1]$ and let $u_i^t = y_i h_t(\mathbf{x}_i)$
   2. Calculate the edge γ_t of h_t: $\gamma_t = \mathbf{d}^t \cdot \mathbf{u}^t$
   3. Set $\widehat{\gamma}_\mathbf{t} = \left(\min_{\mathbf{q}=\mathbf{1},\ldots,\mathbf{t}} \gamma_\mathbf{q} \right) - \delta$ and solve
      $$\mathbf{d}^{t+1} = \operatorname*{argmin}_{\{\mathbf{d} \in \mathcal{P}^N \mid \mathbf{d}\cdot\mathbf{u}^q \leq \widehat{\gamma}_\mathbf{t}, \text{ for } 1 \leq q \leq t\} = \mathcal{C}_t} \Delta(\mathbf{d}, \mathbf{d}^1)$$
   4. **If** above infeasible or $\mathbf{d}^{t+1}$ contains a zero then $T = t$ and break
4. **Output:** $f_\alpha(\mathbf{x}) = \sum_{t=1}^T \alpha_t h_t(\mathbf{x})$, where the coefficients α_t maximize margin over hypotheses set $\{h_1, \ldots, h_T\}$.

TotalBoost$_\delta$

Friedrich Miescher Laboratory
of the Max Planck Society

1. **Input:** $S = \langle (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N) \rangle$, desired accuracy δ
2. **Initialize:** $d_n^1 = 1/N$ for all $n = 1 \ldots N$
3. **Do for** $t = 1, \ldots$
   1. Train classifier on $\{S, \mathbf{d}^t\}$ and obtain hypothesis $h_t : \mathbf{x} \mapsto [-1, 1]$ and let $u_i^t = y_i h_t(\mathbf{x}_i)$
   2. Calculate the edge γ_t of h_t: $\gamma_\mathbf{t} = \mathbf{d^t} \cdot \mathbf{u^t}$
   3. Set $\widehat{\gamma}_\mathbf{t} = \left(\min_{\mathbf{q}=\mathbf{1},\ldots,\mathbf{t}} \gamma_\mathbf{q} \right) - \delta$ and solve
   4. **Optimization Problem**

$$\mathbf{d}^{t+1} = \operatorname*{argmin}_{\mathbf{d} \in \mathcal{C}_t} \Delta(\mathbf{d}, \mathbf{d}^1)$$

$$\text{with} \quad \mathcal{C}_t := \{\mathbf{d} \in \mathcal{P}^N \mid \mathbf{d} \cdot \mathbf{u}^q \leq \widehat{\gamma}_\mathbf{t}, \text{ for } 1 \leq q \leq t\}$$

4. **Output:** $f_\alpha(\mathbf{x}) = \sum_{t=1}^T \alpha_t h_t(\mathbf{x})$, where the coefficients α_t maximize margin over hypotheses set $\{h_1, \ldots, h_T\}$.

Illustrative Experiments

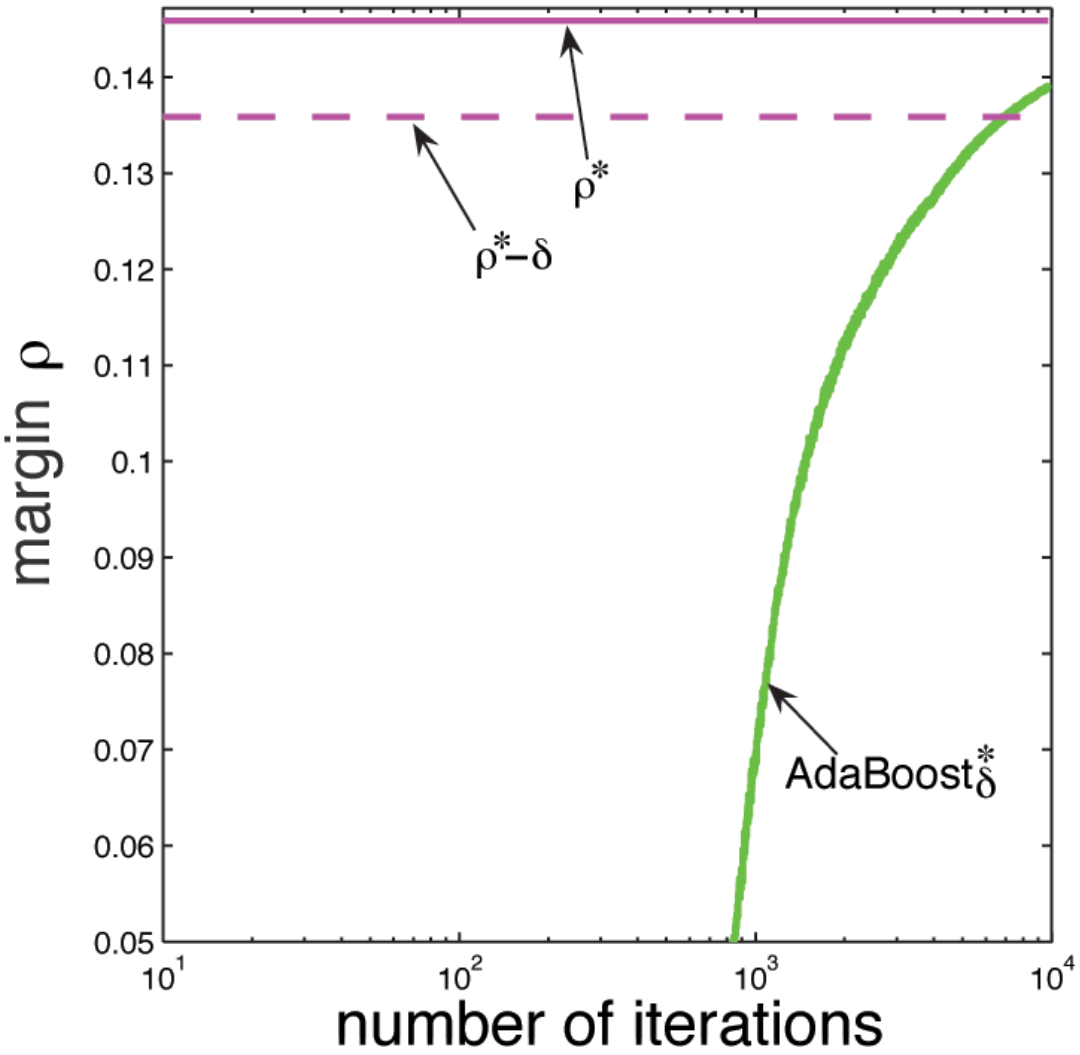
Cox-1 dataset from Telik Inc.

- Relatively small drug-design data set
  - 125 binary labeled examples
  - 3888 binary features
- Compare convergence of margin versus number of iterations

[Warmuth, Liao & G.R., 2006]

Illustrative Experiments

Friedrich Miescher Laboratory
of the Max Planck Society



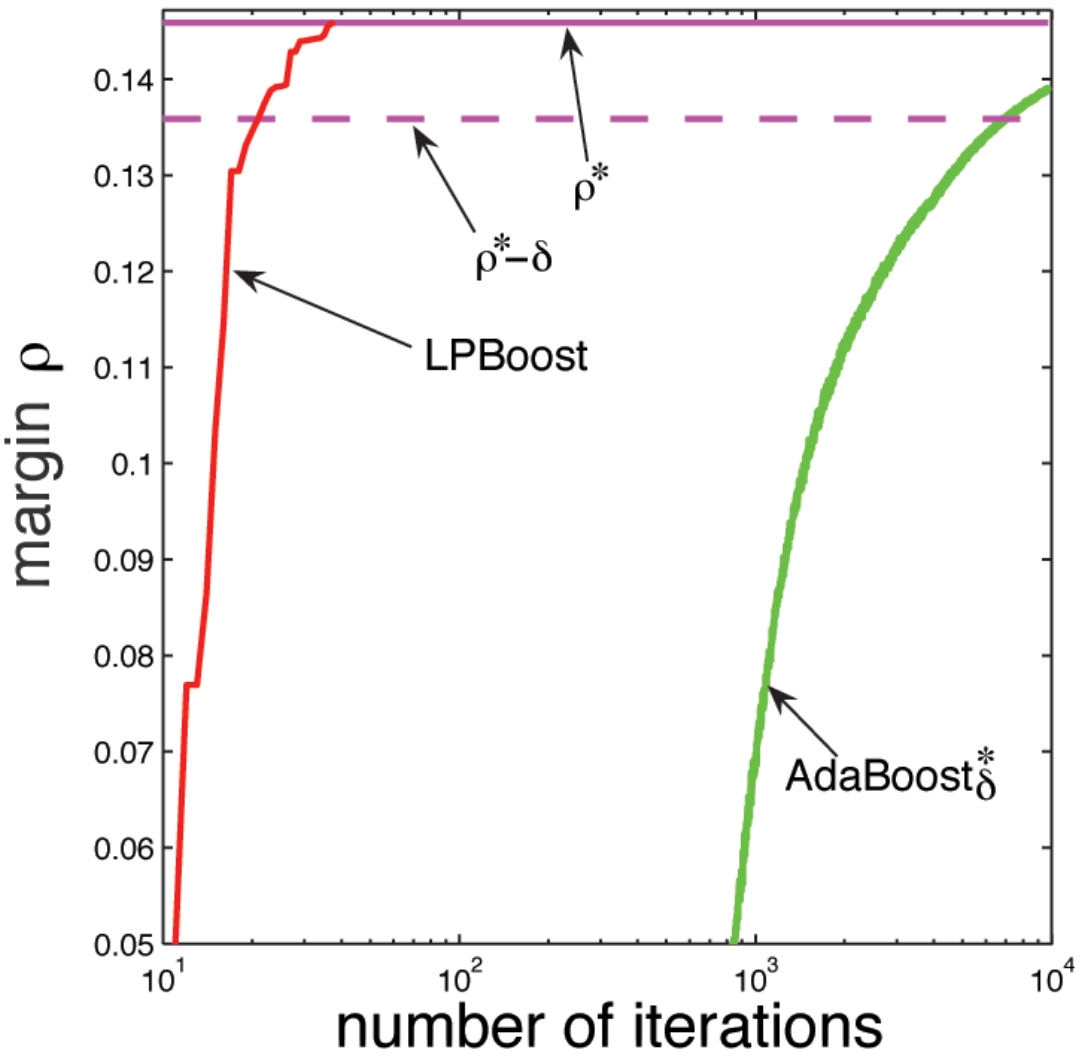
Cox-1 ($\delta = 0.01$)

- AdaBoost$^*_\delta$
- LPBoost
- TotalBoost$_\delta$

Results

Illustrative Experiments

Friedrich Miescher Laboratory
of the Max Planck Society



Cox-1 ($\delta = 0.01$)

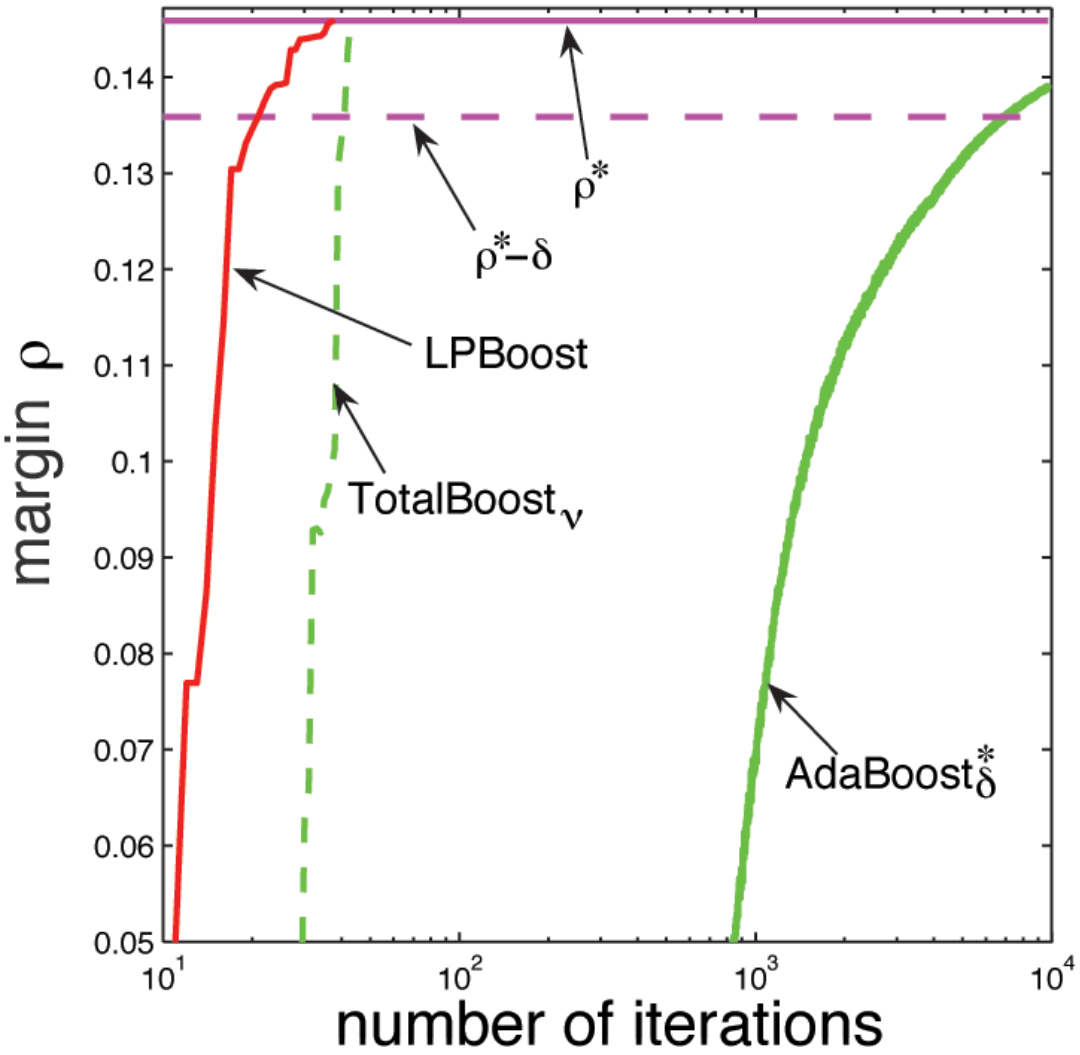
- AdaBoost$^*_\delta$
- LPBoost
- TotalBoost$_\delta$

Results

- Corrective algorithms very slow
- LPBoost & TotalBoost need few iterations
- final predictor sparse (only 100 out of 2200)

Illustrative Experiments

Friedrich Miescher Laboratory
of the Max Planck Society



Cox-1 ($\delta = 0.01$)

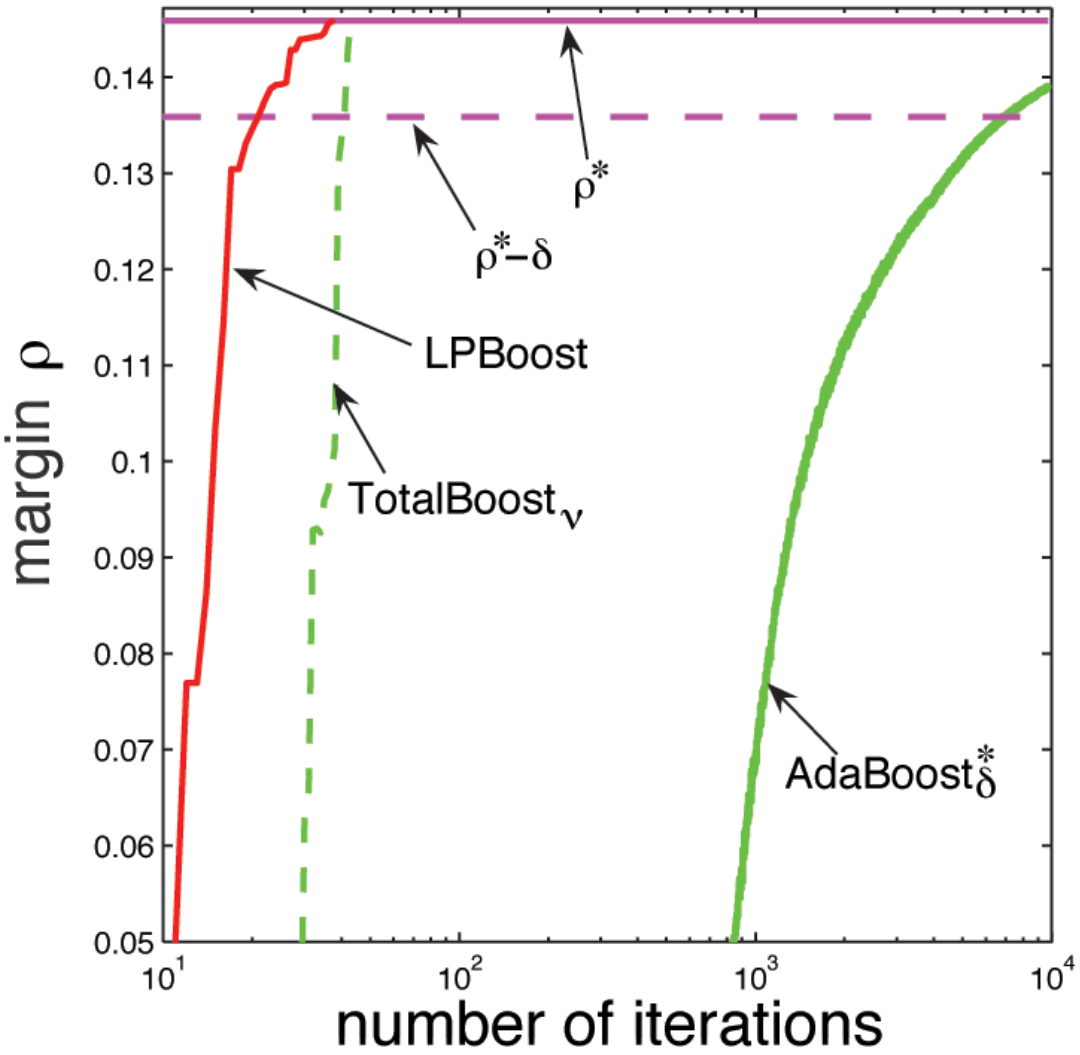
- AdaBoost$^*_\delta$
- LPBoost
- TotalBoost$_\delta$

Results

- Corrective algorithms very slow
- LPBoost & TotalBoost$_\delta$ need few iterations
- Initial speed crucially depends on δ

Illustrative Experiments

Friedrich Miescher Laboratory
of the Max Planck Society



Cox-1 ($\delta = 0.01$)

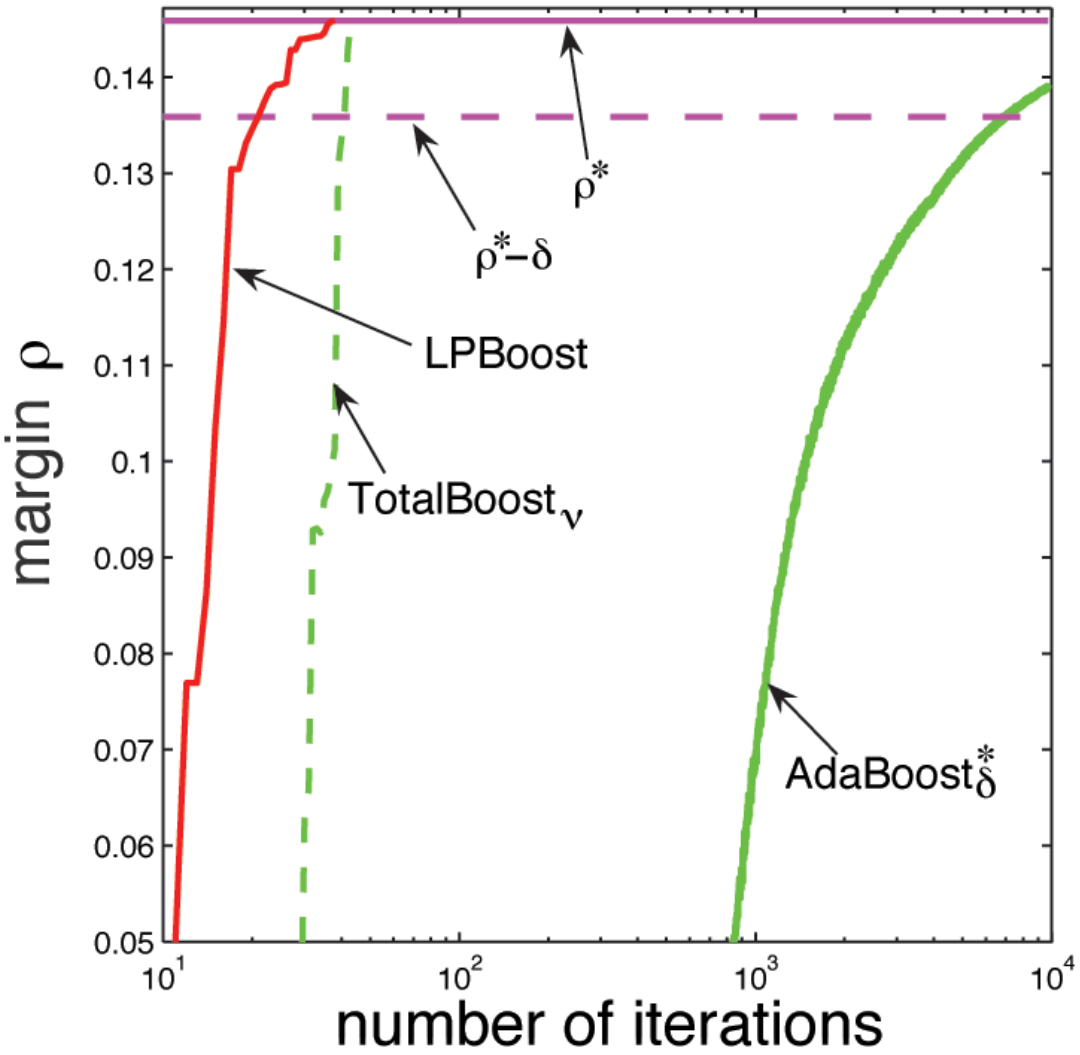
- AdaBoost$^*_\delta$
- LPBoost
- TotalBoost$_\delta$

Results

- Corrective algorithms very slow
- LPBoost & TotalBoost$_\delta$ need few iterations
- Initial speed crucially depends on δ

Illustrative Experiments

Friedrich Miescher Laboratory
of the Max Planck Society



Cox-1 ($\delta = 0.01$)

- AdaBoost$^*_\delta$
- LPBoost
- TotalBoost$_\delta$

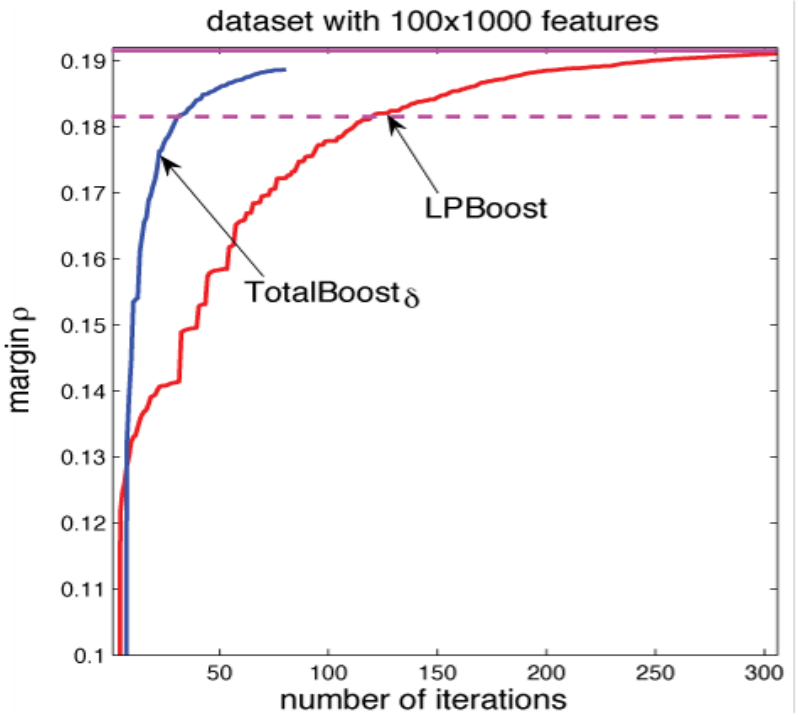
Results

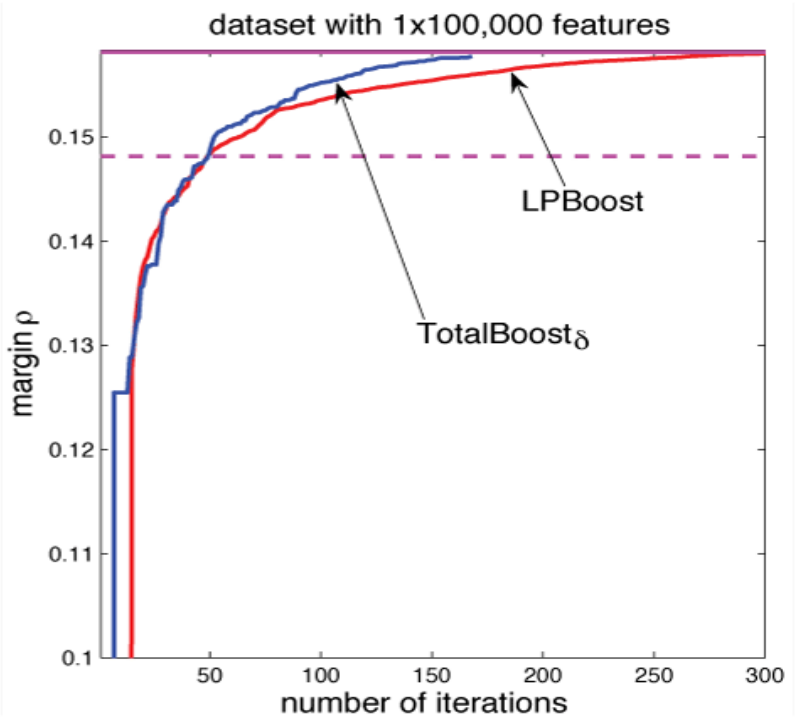
- Corrective algorithms very slow
- LPBoost & TotalBoost$_\delta$ need few iterations
- Initial speed crucially depends on δ

LPBoost May Perform Much Worse Than TotalBoost

- Empirically identified cases where LPBoost converges considerably slower than TotalBoost$_\delta$
- Artificial datasets with 1000 examples:
  - Generated N_1 random ± 1-valued features $x_1, \ldots, x_{N_1}$ and set the label of the examples as $y = \mathrm{sign}(x_1 + x_2 + x_3 + x_4 + x_5)$
  - Duplicated each feature N_2 times, perturbed the features by Gaussian noise ($\sigma = 0.1$), and clipped the feature values so that they lie in the interval $[-1, 1]$
  - Considered two cases:
    1. $N_1 = 1000$, $N_2 = 100$
    2. $N_1 = 100,000$, $N_2 = 1$

LPBoost Worse with Redundant Features





Shown is the margin vs. number of iterations of LPBoost vs. TotalBoost$_\delta$ on two 100,000 dimensional datasets:

left: many redundant features ($N_1 = 1,000, N_2 = 100$) and

right: independent features ($N_1 = 100,000, N_2 = 1$).

Bound Not True for LPBoost

Hypothesis No.						Distribution of Examples Iteration No.				
1	2	3	4	5	6	1	2	3	4	5
+	-	-	-	-	-	1/9	0	0	0	0
+	-	-	-	-	-	1/9	0	0	0	0
+	-	-	-	-	-	1/9	0	0	0	0
+	-	-	-	-	-	1/9	0	0	0	0
+	-	-	-	-	-	1/9	0	0	0	0
-	+	-	-	-	+	1/9	1	0	0	0
-	-	+	-	-	+	1/9	0	1	0	0
-	-	-	+	-	+	1/9	0	0	1	0
-	-	-	-	+	+	1/9	0	0	0	1
						Selected Hypothesis No.				
						1	2	3	4	5

$\Rightarrow$ LPBoost uses 5 iterations ((N+1)/2 iterations) to achieve margin 0.

Theorem [Warmuth, Glocer, Rätsch, 2007]

LPBoost requires $\Omega(N)$ iterations to get close to the maximal margin.

Bound Not True for LPBoost

Hypothesis No.						Distribution of Examples Iteration No.				
1	2	3	4	5	6	1	2	3	4	5
+	-	-	-	-	-	1/9	0	0	0	0
+	-	-	-	-	-	1/9	0	0	0	0
+	-	-	-	-	-	1/9	0	0	0	0
+	-	-	-	-	-	1/9	0	0	0	0
+	-	-	-	-	-	1/9	0	0	0	0
-	+	-	-	-	+	1/9	1	0	0	0
-	-	+	-	-	+	1/9	0	1	0	0
-	-	-	+	-	+	1/9	0	0	1	0
-	-	-	-	+	+	1/9	0	0	0	1
						Selected Hypothesis No.				
						1	2	3	4	5

$\Rightarrow$ LPBoost uses 5 iterations ((N+1)/2 iterations) to achieve margin 0.

Theorem [Warmuth, Glocer, Rätsch, 2007]

LPBoost requires $\Omega(N)$ iterations to get close to the maximal margin.

Bound Not True for LPBoost

Hypothesis No.						Distribution of Examples Iteration No.				
1	2	3	4	5	6	1	2	3	4	5
+	-	-	-	-	-	1/9	0	0	0	0
+	-	-	-	-	-	1/9	0	0	0	0
+	-	-	-	-	-	1/9	0	0	0	0
+	-	-	-	-	-	1/9	0	0	0	0
+	-	-	-	-	-	1/9	0	0	0	0
-	+	-	-	-	+	1/9	1	0	0	0
-	-	+	-	-	+	1/9	0	1	0	0
-	-	-	+	-	+	1/9	0	0	1	0
-	-	-	-	+	+	1/9	0	0	0	1
						Selected Hypothesis No.				
						1	2	3	4	5

$\Rightarrow$ LPBoost uses 5 iterations ((N+1)/2 iterations) to achieve margin 0.

Theorem [Warmuth, Glocer, Rätsch, 2007]

LPBoost requires $\Omega(N)$ iterations to get close to the maximal margin.

Digestion

Primal

$$\max_{\boldsymbol{\alpha},\rho} \quad \rho$$

$$\text{with} \quad \sum_{h\in\mathcal{H}} \alpha_h y_n h(\mathbf{x}_n) \geq \rho \quad \forall n$$

$$\boldsymbol{\alpha} \in \mathcal{P}^{|\mathcal{H}|}$$

Dual

$$\min_{\gamma,\mathbf{d}} \quad \gamma$$

$$\text{with} \quad \sum_{n=1}^{N} d_n y_n h(\mathbf{x}_n) \leq \gamma \quad \forall h \in \mathcal{H}$$

$$\mathbf{d} \in \mathcal{P}^{N}$$

Infinite number of hypotheses: $|\mathcal{H}| \to \infty$

- Dual has infinitely many constraints, but still well defined
- There exist solutions with at most N active variables/constraints
- **Convergence theorems still hold**

$\Rightarrow$ Solution to "Semi-infinite Linear Program"

[Rätsch, Demiriz & Bennett, 2002, Rätsch, 2003]

Digestion

Primal

$$\max_{\boldsymbol{\alpha},\rho} \quad \rho$$

$$\text{with} \quad \sum_{h\in\mathcal{H}} \alpha_h y_n h(\mathbf{x}_n) \geq \rho \quad \forall n$$

$$\boldsymbol{\alpha} \in \mathcal{P}^{|\mathcal{H}|}$$

Dual

$$\min_{\gamma,\mathbf{d}} \quad \gamma$$

$$\text{with} \quad \sum_{n=1}^{N} d_n y_n h(\mathbf{x}_n) \leq \gamma \quad \forall h \in \mathcal{H}$$

$$\mathbf{d} \in \mathcal{P}^{N}$$

Infinite number of hypotheses: $|\mathcal{H}| \to \infty$

- Dual has infinitely many constraints, but still well defined
- There exist solutions with at most N active variables/constraints
- **Convergence theorems still hold**

$\Rightarrow$ Solution to "Semi-infinite Linear Program"

[Rätsch, Demiriz & Bennett, 2002; Rätsch, 2003]

Digestion

Primal

$$\max_{\boldsymbol{\alpha},\rho} \quad \rho$$

$$\text{with} \quad \sum_{h\in\mathcal{H}} \alpha_h y_n h(\mathbf{x}_n) \geq \rho \quad \forall n$$

$$\boldsymbol{\alpha} \in \mathcal{P}^{|\mathcal{H}|}$$

Dual

$$\min_{\gamma,\mathbf{d}} \quad \gamma$$

$$\text{with} \quad \sum_{n=1}^{N} d_n y_n h(\mathbf{x}_n) \leq \gamma \quad \forall h \in \mathcal{H}$$

$$\mathbf{d} \in \mathcal{P}^{N}$$

Infinite number of hypotheses: $|\mathcal{H}| \rightarrow \infty$

- Dual has infinitely many constraints, but still well defined
- There exist solutions with at most N active variables/constraints
- **Convergence theorems still hold**

$\Rightarrow$ Solution to "Semi-infinite Linear Program"

JMLR - Machine Learning Open Source Publications

`http://jmlr.csail.mit.edu/mloss`

To support the open source movement, JMLR has announced a way of publishing open source software. Submissions should include:

- Four page description
- Code
- Recognised open source license

which are contributions related to:

- Implementations of machine learning algorithms
- Toolboxes and languages for scientific computing

See also position statement in "The Need for Open Source Software in Machine Learning" by Sonnenburg et al., JMLR, 2007

Central Repository: `http://www.mloss.org`

Do These Algorithms Perform Better in Practice?

Do they generalize better?

- Algorithms converge faster, but often do not generalize better.
- Only slight improvements in the noise free case.

Soft margins for AdaBoost

1. Limit the *influence* of examples ("AdaBoost$_{\text{reg}}$")
   - Heuristic algorithm, no convergence result ...
   - ... but works very well in practice
2. Soft margins à la SVM ("ν-Arc", "SoftBoost")
   - Change optimization problem to include slacks
   - Convergence proofs did not work $\Rightarrow$ only asymptotic results

[G.R. et al. 2000-2005, Warmuth, Glocer & G.R., 2007]

Do These Algorithms Perform Better in Practice?

Do they generalize better?

- Algorithms converge faster, but often do not generalize better.
- Only slight improvements in the noise free case.

Soft margins for AdaBoost

1. Limit the *influence* of examples ("AdaBoost$_{\text{reg}}$")
   - Heuristic algorithm, no convergence result ...
   - ... but works very well in practice
2. Soft margins à la SVM ("ν-Arc", "SoftBoost")
   - Change optimization problem to include slacks
   - Convergence proofs did not work $\Rightarrow$ only asymptotic results
   - New: Can prove fast convergence with soft margin (NIPS'07)

[G.R. et al. 2000-2005, Warmuth, Glocer & G.R., 2007]

Do These Algorithms Perform Better in Practice?

Do they generalize better?

- Algorithms converge faster, but often do not generalize better.
- Only slight improvements in the noise free case.

Soft margins for AdaBoost

1. Limit the *influence* of examples ("$\text{AdaBoost}_{\text{reg}}$")
   - Heuristic algorithm, no convergence result ...
   - ... but works very well in practice
2. Soft margins à la SVM ("ν-Arc", "SoftBoost")
   - Change optimization problem to include slacks
   - Convergence proofs did not work $\Rightarrow$ only asymptotic results
   - **New:** Can prove fast convergence with soft margin (NIPS'07)

[G.R. et al. 2000-2005, Warmuth, Glocer & G.R., 2007]

Do These Algorithms Perform Better in Practice?

Do they generalize better?

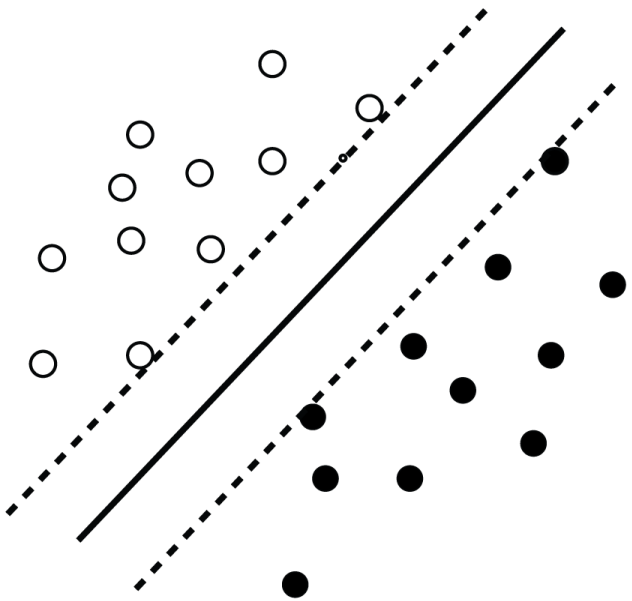
- Algorithms converge faster, but often do not generalize better.
- Only slight improvements in the noise free case.

Soft margins for AdaBoost

1. Limit the *influence* of examples ("AdaBoost$_{\text{reg}}$")
   - Heuristic algorithm, no convergence result ...
   - ... but works very well in practice
2. Soft margins à la SVM ("ν-Arc", "SoftBoost")
   - Change optimization problem to include slacks
   - Convergence proofs did not work $\Rightarrow$ only asymptotic results
   - **New:** Can prove fast convergence with soft margin (NIPS'07)

[G.R. et al. 2000-2005, Warmuth, Glocer & G.R., 2007]

SoftBoost = TotalBoost with ν-Trick



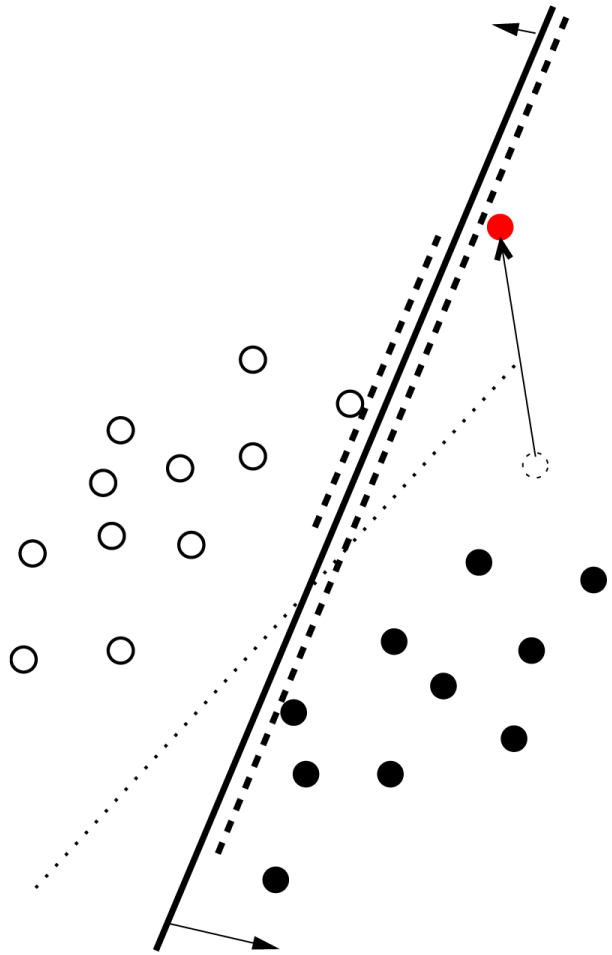
Hard-margin separation

$$\max_{\rho,\boldsymbol{\xi},\boldsymbol{\alpha}} \quad \rho - \frac{1}{\nu}\sum_{n=1}^{N}\xi_n$$

$$\text{with} \quad y_n \sum_h \alpha_h h(\mathbf{x}_n) \geq \rho - \xi_n \quad \text{for } n = 1,\ldots,N.$$

$$\text{s.t.} \quad \rho \geq 0, \boldsymbol{\xi} \in \mathbb{R}_+, \boldsymbol{\alpha} \in \mathcal{P}^{|\mathcal{H}|}$$

SoftBoost = TotalBoost with ν-Trick



$$\max_{\rho,\boldsymbol{\xi},\boldsymbol{\alpha}} \quad \rho - \frac{1}{\nu}\sum_{n=1}^{N}\xi_n$$

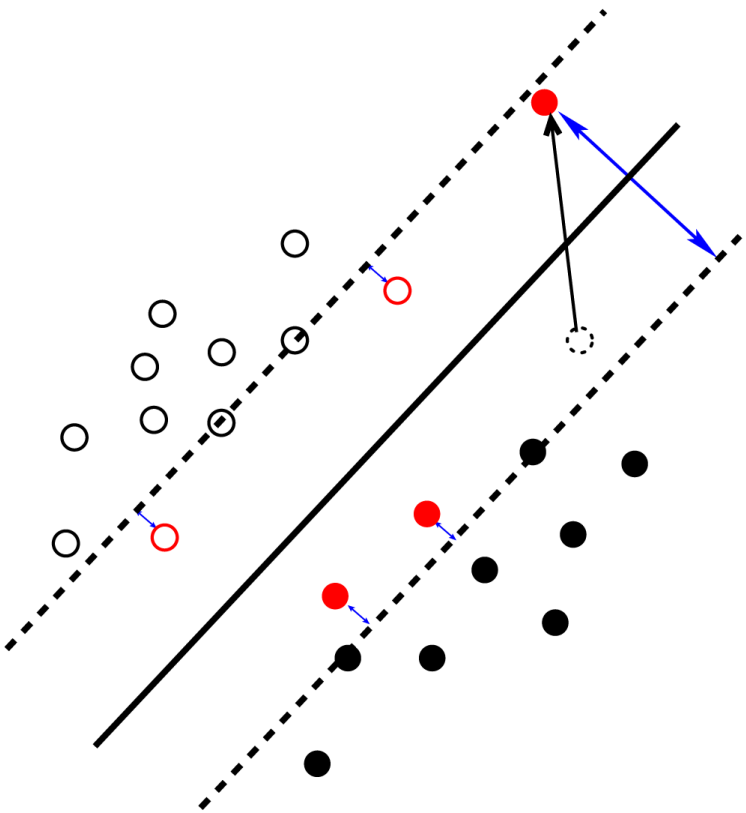
$$\text{with} \quad y_n \sum_h \alpha_h h(\mathbf{x}_n) \geq \rho - \xi_n$$

$$\text{for } n = 1,\ldots,N.$$

$$\text{s.t.} \quad \rho \geq 0, \boldsymbol{\xi} \in \mathbb{R}_+, \boldsymbol{\alpha} \in \mathcal{P}^{|\mathcal{H}|}$$

Hard-margin separation with outlier

SoftBoost = TotalBoost with ν-Trick



$$\max_{\rho,\boldsymbol{\xi},\boldsymbol{\alpha}} \quad \rho - \frac{1}{\nu}\sum_{n=1}^{N}\xi_n$$

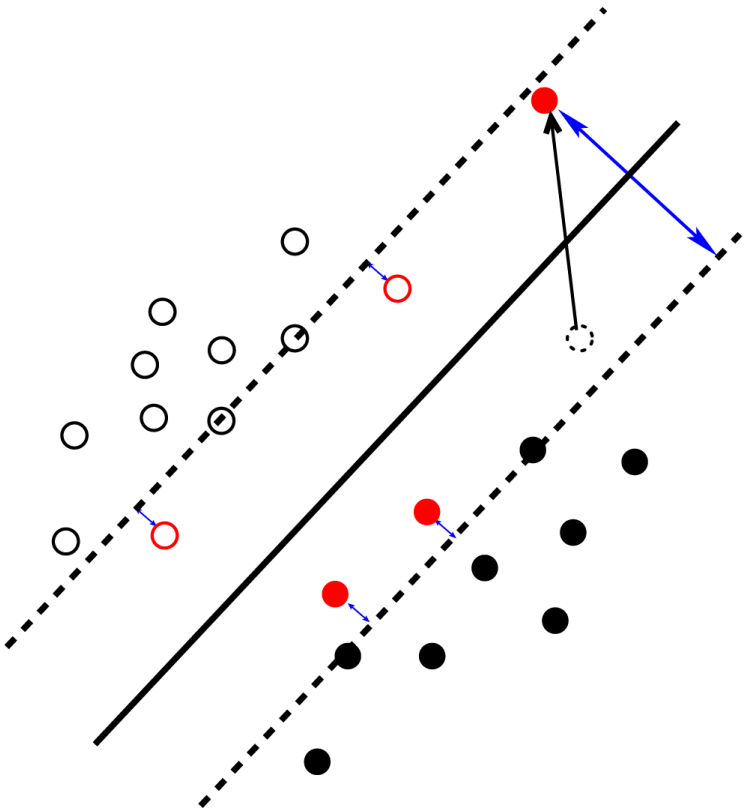
$$\text{with} \quad y_n \sum_h \alpha_h h(\mathbf{x}_n) \geq \rho - \xi_n \quad \text{for } n = 1,\ldots,N.$$

$$\text{s.t.} \quad \rho \geq 0, \boldsymbol{\xi} \in \mathbb{R}_+, \boldsymbol{\alpha} \in \mathcal{P}^{|\mathcal{H}|}$$

Soft-margin separation with outlier

Chose margin such that ν examples are within the margin area

SoftBoost = TotalBoost with ν-Trick



$$\max_{\rho,\boldsymbol{\xi},\boldsymbol{\alpha}} \quad \rho - \frac{1}{\nu}\sum_{n=1}^{N}\xi_n$$

$$\text{with} \quad y_n \sum_h \alpha_h h(\mathbf{x}_n) \geq \rho - \xi_n$$

$$\text{for } n = 1, \ldots, N.$$

$$\text{s.t.} \quad \rho \geq 0, \boldsymbol{\xi} \in \mathbb{R}_+, \boldsymbol{\alpha} \in \mathcal{P}^{|\mathcal{H}|}$$

Soft-margin separation with outlier

Chose margin such that ν examples are within the margin area

Softboost$_{\delta,\nu}$

1. **Input:** $S = \langle (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N) \rangle$, desired accuracy δ
2. **Initialize:** $d_n^1 = 1/N$ for all $n = 1 \ldots N$
3. **Do for** $t = 1, \ldots$
   1. Train classifier on $\{S, \mathbf{d}^t\}$ and obtain hypothesis $h_t : \mathbf{x} \mapsto [-1, 1]$ and let $u_i^t = y_i h_t(\mathbf{x}_i)$
   2. Calculate the edge γ_t of h_t: $\gamma_\mathbf{t} = \mathbf{d^t} \cdot \mathbf{u^t}$
   3. Set $\widehat{\gamma}_\mathbf{t} = \left(\min_{\mathbf{q}=\mathbf{1},\ldots,\mathbf{t}} \gamma_\mathbf{q} \right) - \delta$ and solve
      $$\mathbf{d}^{t+1} = \operatorname*{argmin}_{\{\mathbf{d} \in \mathcal{P}^N \mid \mathbf{d} \leq 1/\nu; \mathbf{d}\cdot\mathbf{u}^q \leq \widehat{\gamma}_\mathbf{t}, \text{ for } 1 \leq q \leq t\} = \mathcal{C}_t} \Delta(\mathbf{d}, \mathbf{d}^1)$$
   4. **If** above infeasible or $\mathbf{d}^{t+1}$ contains a zero then $T = t$ and break
4. **Output:** $f_\alpha(\mathbf{x}) = \sum_{t=1}^T \alpha_t h_t(\mathbf{x})$, where the coefficients α_t maximize margin over hypotheses set $\{h_1, \ldots, h_T\}$.

Softboost$_{\delta,\nu}$

1. **Input:** $S = \langle (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N) \rangle$, desired accuracy δ
2. **Initialize:** $d_n^1 = 1/N$ for all $n = 1 \ldots N$
3. **Do for** $t = 1, \ldots$
   1. Train classifier on $\{S, \mathbf{d}^t\}$ and obtain hypothesis $h_t : \mathbf{x} \mapsto [-1, 1]$ and let $u_i^t = y_i h_t(\mathbf{x}_i)$
   2. Calculate the edge γ_t of h_t: $\gamma_\mathbf{t} = \mathbf{d^t} \cdot \mathbf{u^t}$
   3. Set $\widehat{\gamma}_\mathbf{t} = (\min_{\mathbf{q}=\mathbf{1},\ldots,\mathbf{t}} \gamma_\mathbf{q}) - \delta$ and solve

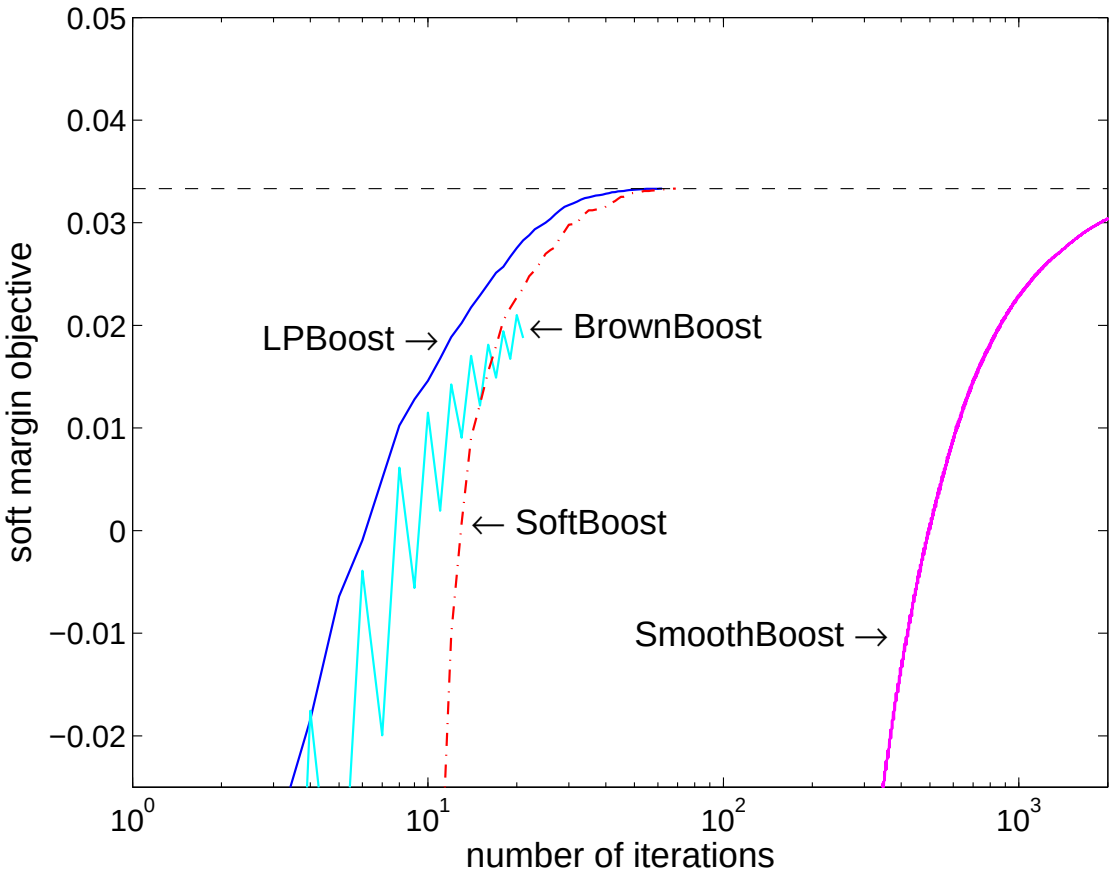
Optimization Problem

$$\mathbf{d}^{t+1} = \operatorname*{argmin}_{\mathbf{d} \in \mathcal{C}_t} \Delta(\mathbf{d}, \mathbf{d}^1)$$

with $$\mathcal{C}_t := \{\mathbf{d} \in \mathcal{P}^N \mid \mathbf{d} \leq 1/\nu,\ \mathbf{d} \cdot \mathbf{u}^q \leq \widehat{\gamma}_\mathbf{t},\ \text{for } 1 \leq q \leq t\}$$

4. **Output:** $f_\alpha(\mathbf{x}) = \sum_{t=1}^T \alpha_t h_t(\mathbf{x})$, where the coefficients α_t maximize margin over hypotheses set $\{h_1, \ldots, h_T\}$.

Convergence Speed of Different Algorithms



Soft margin objective vs. the number of iterations for LPBoost, SoftBoost, BrownBoost and SmoothBoost.

[Freund, 2001, Demiriz et al., 2002, Servedio, 2003, Warmuth, Glocer & G.R., 2007]

Generalization Errors on IDA Benchmarks

- RBF networks as base learner
- 5-fold cross-validation on 100 splits
- Average error ± standard deviation
- LPBoost very similar to SoftBoost (not shown)

	AdaBoost	BrownBoost	AdaBoost reg	SoftBoost
Banana	13.3 ± 0.7	12.9 ± 0.7	11.3 ± 0.6	**11.1 ± 0.5**
B.Cancer	32.1 ± 3.8	30.2 ± 3.9	**27.3 ± 4.3**	28.0 ± 4.5
Diabetes	27.9 ± 1.5	27.2 ± 1.6	24.5 ± 1.7	**24.4 ± 1.7**
German	26.9 ± 1.9	24.8 ± 1.9	25.0 ± 2.2	**24.7 ± 2.1**
Heart	20.1 ± 2.7	20.0 ± 2.8	**17.6 ± 3.0**	18.2 ± 2.7
Ringnorm	1.9 ± 0.3*	1.9 ± 0.2	**1.7 ± 0.2**	1.8 ± 0.2
F.Solar	36.1 ± 1.5	36.1 ± 1.4	**34.4 ± 1.7**	35.5 ± 1.4
Thyroid	**4.4 ± 1.9***	4.6 ± 2.1	4.9 ± 2.0	4.9 ± 1.9
Titanic	22.8 ± 1.0	22.8 ± 0.8	**22.7 ± 1.0**	23.0 ± 0.8
Waveform	10.5 ± 0.4	10.4 ± 0.4	10.4 ± 0.7	**9.8 ± 0.5**

[Warmuth, Glocer & G.R., 2007]

So, All Work is Done!?

- SoftBoost combines all benefits [Warmuth, Glocer, G.R., NIPS, 2007]
  - Noise robustness
  - Good rates of convergence
  - Few iterations in practice
- Let's apply it!
  - Benchmarks
  - Drug Discovery (with K. Bennett, K.-R. Müller)
  - Electric appliances monitoring (with T. Onoda)
    - Non-intrusively classify the on/off state of appliances
    - Patented and in use in Japan
  - Biological sequence analysis
    - Classify sequences (e.g. exons vs. introns)
    - Features correspond to subsequences
    - Boosting/LP based methods suffer in practical cases
    - SVMs!

So, All Work is Done!?

- SoftBoost combines all benefits [Warmuth, Glocer, G.R., NIPS, 2007]
  - Noise robustness
  - Good rates of convergence
  - Few iterations in practice
- Let's apply it!
  - \+ Benchmarks
  - \+ Drug Discovery (with K. Bennett, K.-R. Müller)
  - \+ Electric appliances monitoring (with T. Onoda)
    - Non-intrusively classify the on/off state of appliances
    - Patented and in use in Japan
  - − Biological sequence analysis
    - Classify sequences (e.g. exons vs. introns)
    - Features correspond to subsequences
    - Boosting/LP based methods fail in practical cases
    - **Why?**

So, All Work is Done!?

- SoftBoost combines all benefits [Warmuth, Glocer, G.R., NIPS, 2007]
  - Noise robustness
  - Good rates of convergence
  - Few iterations in practice
- Let's apply it!
  - \+ Benchmarks
  - \+ Drug Discovery (with K. Bennett, K.-R. Müller)
  - \+ Electric appliances monitoring (with T. Onoda)
    - Non-intrusively classify the on/off state of appliances
    - Patented and in use in Japan
  - − Biological sequence analysis
    - Classify sequences (e.g. exons vs. introns)
    - Features correspond to subsequences
    - Boosting/LP based methods fail in practical cases
    - **Why?**

So, All Work is Done!?

- SoftBoost combines all benefits [Warmuth, Glocer, G.R., NIPS, 2007]
  - Noise robustness
  - Good rates of convergence
  - Few iterations in practice
- Let's apply it!
  - \+ Benchmarks
  - \+ Drug Discovery (with K. Bennett, K.-R. Müller)
  - \+ Electric appliances monitoring (with T. Onoda)
    - Non-intrusively classify the on/off state of appliances
    - Patented and in use in Japan
  - – Biological sequence analysis
    - Classify sequences (e.g. exons vs. introns)
    - Features correspond to subsequences
    - Boosting/LP based methods fail in practical cases
    - **Why?**

So, All Work is Done!?

- SoftBoost combines all benefits [Warmuth, Glocer, G.R., NIPS, 2007]
  - Noise robustness
  - Good rates of convergence
  - Few iterations in practice
- Let's apply it!
  - \+ Benchmarks
  - \+ Drug Discovery (with K. Bennett, K.-R. Müller)
  - \+ Electric appliances monitoring (with T. Onoda)
    - Non-intrusively classify the on/off state of appliances
    - Patented and in use in Japan
  - – Biological sequence analysis
    - Classify sequences (e.g. exons vs. introns)
    - Features correspond to subsequences
    - Boosting/LP based methods fail in practical cases
    - **Why?**

Illustrative Example: Classify Sequences

Friedrich Miescher Laboratory
of the Max Planck Society

Task 1

- *Single substring* discriminates, e.g. `GATTACA`
- Optimal weight vector **sparse**
- Results
  - \+ Boosting/LP
  - − SVM
- Sparseness beneficial
  - Quick identification of discriminating dimensions

Task 2

- Differing *substring frequencies*
- Optimal weight vector **dense**
- Results
  - − Boosting/LP
  - \+ SVM
- Sparseness disadvantageous
  - Concentrates on too few dimensions

Discrimination of biological signals' sequences

- Needs many different features
- Kernel methods often work better

Illustrative Example: Classify Sequences

Friedrich Miescher Laboratory
of the Max Planck Society

Task 1

- *Single substring* discriminates, e.g. `GATTACA`
- Optimal weight vector **sparse**
- Results
  - \+ Boosting/LP
  - − SVM
- Sparseness beneficial
  - Quick identification of discriminating dimensions

Task 2

- Differing *substring frequencies*
- Optimal weight vector **dense**
- Results
  - − Boosting/LP
  - \+ SVM
- Sparseness detrimental
  - Concentration on too few dimensions

Discrimination of biological signals' sequences

- Needs many different features
- Kernel methods often work better

Illustrative Example: Classify Sequences

Friedrich Miescher Laboratory
of the Max Planck Society

Task 1

- *Single substring* discriminates, e.g. `GATTACA`
- Optimal weight vector **sparse**
- Results
  - \+ Boosting/LP
  - – SVM
- Sparseness beneficial
  - Quick identification of discriminating dimensions

Task 2

- Differing *substring frequencies*
- Optimal weight vector **dense**
- Results
  - – Boosting/LP
  - \+ SVM
- Sparseness detrimental
  - Concentration on too few dimensions

Discrimination of biological signals' sequences

- Needs many different features
- Kernel methods often work better

Illustrative Example: Classify Sequences

Friedrich Miescher Laboratory
of the Max Planck Society

Task 1

- *Single substring* discriminates, e.g. `GATTACA`
- Optimal weight vector **sparse**
- Results
  - \+ Boosting/LP
  - – SVM
- Sparseness beneficial
  - Quick identification of discriminating dimensions

Task 2

- Differing *substring frequencies*
- Optimal weight vector **dense**
- Results
  - – Boosting/LP
  - \+ SVM
- Sparseness detrimental
  - Concentration on too few dimensions

Discrimination of biological signals' sequences

- Needs many different features
- Kernel methods often work better

Illustrative Example: Classify Sequences

Friedrich Miescher Laboratory
of the Max Planck Society

Task 1

- *Single substring* discriminates, e.g. `GATTACA`
- Optimal weight vector **sparse**
- Results
  - \+ Boosting/LP
  - – SVM
- Sparseness beneficial
  - Quick identification of discriminating dimensions

Task 2

- Differing *substring frequencies*
- Optimal weight vector **dense**
- Results
  - – Boosting/LP
  - \+ SVM
- Sparseness detrimental
  - Concentration on too few dimensions

Discrimination of biological signals' sequences

- Needs many different features
- Kernel methods often work better

Combination: Multiple Kernel Learning

- Kernel $k(\mathbf{x}, \mathbf{x}') = \Phi(\mathbf{x}) \cdot \Phi(\mathbf{x}')$ used in standard SVM Classifier

$$f(\mathbf{x}) = \sum_{i=1}^{\ell} y_i \alpha_i \mathsf{k}(\mathbf{x}, \mathbf{x}_i) + b$$

- Now: Convex combination of kernels

$$\mathsf{k}(\mathbf{x}, \mathbf{x}') = \sum_{j=1}^{M} \beta_j \, \mathsf{k}_j(\mathbf{x}, \mathbf{x}'), \qquad \boldsymbol{\beta} \in \mathcal{P}^M$$

e.g. polynomial kernels of different degree, kernels on different domains or heterogeneous data.

$\Rightarrow$ Combination of benefits of sparse and dense learning!

- **But:** How to learn weights β_j ?

MKL Optimization: Reformulation as SILP



$$\begin{aligned} \min \quad & \theta \\ \text{with} \quad & \sum_{j=1}^{M} \beta_j S_j(\boldsymbol{\alpha}) \leq \theta \text{ for all } \boldsymbol{\alpha} \in \mathcal{S}_C \\ \text{s.t.} \quad & \theta \in \mathbf{R}, \boldsymbol{\beta} \in \mathcal{P}^M \end{aligned}$$

Use Boosting techniques to solve SILP

- Exponential update of the β's
- Finding corresponds to finding active constraint:

$$\sum_{j=1}^{M} \beta_j S_j(\boldsymbol{\alpha}) = \frac{1}{2} \sum_{r=1}^{N} \sum_{s=1}^{N} \alpha_r \alpha_s y_r y_s \sum_{j=1}^{M} \beta_j k_j(\mathbf{x}_r, \mathbf{x}_s) - \sum_{i=1}^{N} \alpha_i$$

$\Rightarrow$ **Solve SVM** for some given β's

[Sonnenburg, G.R., Schäfer, 2005]

MKL Optimization: Reformulation as SILP

$$\begin{aligned} \min \quad & \theta \\ \text{with} \quad & \sum_{j=1}^{M} \beta_j S_j(\boldsymbol{\alpha}) \leq \theta \text{ for all } \boldsymbol{\alpha} \in \mathcal{S}_C \\ \text{s.t.} \quad & \theta \in \mathbf{R}, \boldsymbol{\beta} \in \mathcal{P}^M \end{aligned}$$

Use Boosting techniques to solve SILP

- Exponential update of the $\boldsymbol{\beta}$'s
- Finding corresponds to finding active constraint:

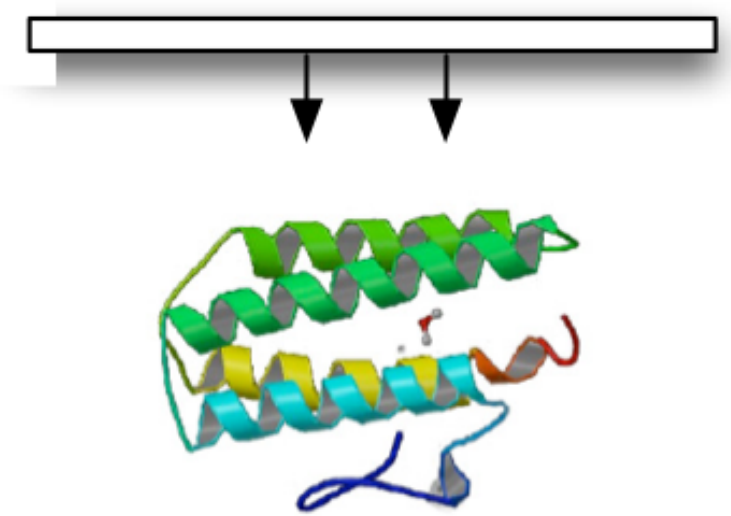
$$\sum_{j=1}^{M} \beta_j S_j(\boldsymbol{\alpha}) = \frac{1}{2} \sum_{r=1}^{N} \sum_{s=1}^{N} \alpha_r \alpha_s y_r y_s \sum_{j=1}^{M} \beta_j k_j(\mathbf{x}_r, \mathbf{x}_s) - \sum_{i=1}^{N} \alpha_i$$

$\Rightarrow$ **Solve SVM** for some given $\boldsymbol{\beta}$'s

[Sonnenburg, G.R., Schäfer, 2005]

Another Case: Learning Structured Outputs

- Learn function S_θ to score "structures" **s** given input **x**



amino-acid sequence

protein structure

[Tsochantaridis et al., 2004]

Another Case: Learning Structured Outputs

- Learn function S_θ to score "structures" $\mathbf{s}$ given input $\mathbf{x}$
- **Idea:** Score true structure much higher than all other structures
- Solve optimization problem:

$$\begin{aligned} \underset{\xi \in \mathbb{R}^N_+, \theta \in \mathbb{R}^d}{\text{minimize}} \quad & C \sum_{n=1}^{N} \xi_n + \|\theta\|_1 \\ \text{subject to} \quad & S_\theta(\mathbf{s}_n^+, \mathbf{x}_n) - S_\theta(\mathbf{s}^-, \mathbf{x}_n) \geq 1 - \xi_n \\ \text{for all} \quad & \mathbf{s}^- \neq \mathbf{s}_n^+ \text{ and } n = 1, \ldots, N \end{aligned}$$

  $\Rightarrow$ Exponentially many constraints

  $\Rightarrow$ Find violating constraints by **dynamic programming**

- Many applications in computational biology
  - Structure prediction
  - Gene finding
  - ...

[Tsochantaridis et al., 2004]

Another Case: Learning Structured Outputs

- Learn function S_θ to score "structures" $\mathbf{s}$ given input $\mathbf{x}$
- **Idea:** Score true structure much higher than all other structures
- Solve optimization problem:

$$\begin{aligned} \underset{\xi \in \mathbb{R}^N_+, \theta \in \mathbb{R}^d}{\text{minimize}} \quad & C \sum_{n=1}^{N} \xi_n + \|\theta\|_1 \\ \text{subject to} \quad & S_\theta(\mathbf{s}_n^+, \mathbf{x}_n) - S_\theta(\mathbf{s}^-, \mathbf{x}_n) \geq 1 - \xi_n \\ \text{for all} \quad & \mathbf{s}^- \neq \mathbf{s}_n^+ \text{ and } n = 1, \ldots, N \end{aligned}$$

  $\Rightarrow$ Exponentially many constraints

  $\Rightarrow$ Find violating constraints by **dynamic programming**
- Many applications in computational biology
  - Structure prediction
  - Gene finding
  - ...

[Tsochantaridis et al., 2004]

Summary

- AdaBoost-type algorithms attempt to achieve a large margin
- TotalBoost/SoftBoost solves SILPs and combine benefits:
  - much faster than AdaBoost, about as fast as LPBoost
  - Convergence guarantee: $\mathcal{O}(\ln N/\delta^2)$
  - State-of-the-art performance
- Huge linear programs frequently appear in machine learning
- Iterative methods require algorithms to identify violated constraints
  - Boosting: **Weak learner**
  - Multiple kernel learning: **Single kernel SVM**
  - Structure Learning: **Dynamic programming**
- Open problems
  - Is there a lower bound or a faster algorithm?
  - Remove dependence on δ? [Warmuth, Glocer, Vishwanathan, 2008]

Summary

- AdaBoost-type algorithms attempt to achieve a large margin
- TotalBoost/SoftBoost solves SILPs and combine benefits:
  - much faster than AdaBoost, about as fast as LPBoost
  - Convergence guarantee: $\mathcal{O}(\ln N/\delta^2)$
  - State-of-the-art performance
- Huge linear programs frequently appear in machine learning
- Iterative methods require algorithms to identify violated constraints
  - Boosting: **Weak learner**
  - Multiple kernel learning: **Single kernel SVM**
  - Structure Learning: **Dynamic programming**
- Open problems
  - Is there a lower bound or a faster algorithm?
  - Remove dependence on δ? [Warmuth, Glocer, Vishwanathan, 2008]

Summary

- AdaBoost-type algorithms attempt to achieve a large margin
- TotalBoost/SoftBoost solves SILPs and combine benefits:
  - much faster than AdaBoost, about as fast as LPBoost
  - Convergence guarantee: $\mathcal{O}(\ln N/\delta^2)$
  - State-of-the-art performance
- Huge linear programs frequently appear in machine learning
- Iterative methods require algorithms to identify violated constraints
  - Boosting: **Weak learner**
  - Multiple kernel learning: **Single kernel SVM**
  - Structure Learning: **Dynamic programming**
- Open problems
  - Is there a lower bound or a faster algorithm?
  - Remove dependence on δ? [Warmuth, Glocer, Vishwanathan, 2008]

Summary

- AdaBoost-type algorithms attempt to achieve a large margin
- TotalBoost/SoftBoost solves SILPs and combine benefits:
  - much faster than AdaBoost, about as fast as LPBoost
  - Convergence guarantee: $\mathcal{O}(\ln N/\delta^2)$
  - State-of-the-art performance
- Huge linear programs frequently appear in machine learning
- Iterative methods require algorithms to identify violated constraints
  - Boosting: **Weak learner**
  - Multiple kernel learning: **Single kernel SVM**
  - Structure Learning: **Dynamic programming**
- Open problems
  - Is there a lower bound or a faster algorithm?
  - Remove dependence on δ? [Warmuth, Glocer, Vishwanathan, 2008]

The End

- Slides with references will be available at `http://www.fml.mpg.de/raetsch/lectures/benelearn08`
- Collaborators for this work:
  Kristin Bennett, Ayhan Demiriz, Karen Glocer, Jun Liao, Sebastian Mika, Klaus-R. Müller, Takashi Onoda, Christin Schäfer, Bernhard Schölkopf, Sören Sonnenburg, **Manfred Warmuth**
- **Thank You!**

References I

Y. Altun, I. Tsochantaridis, and T. Hofmann. Hidden Markov support vector machines. In *Proc. ICML'03*, pages 3–10. AAAI Press, 2003. ISBN 1-57735-189-4.

F. Bach, G. Lanckriet, and M. Jordan. Multiple kernel learning, conic duality, and the SMO algorithm. In C. E. Brodley, editor, *Proc. ICML'04*. ACM, 2004.

L. Breiman. Prediction games and arcing algorithms. *Neural Computation*, 11(7):1493–1518, 1999.

C. Cortes and V.N. Vapnik. Support vector networks. *Machine Learning*, 20:273–297, 1995.

A. Demiriz, K.P. Bennett, and J. Shawe-Taylor. Linear programming boosting via column generation. *Machine Learning*, 46:225–254, 2002.

References II

Y. Freund. An adaptive version of the boost by majority algorithm. *Mach. Learn.*, 43(3):293–318, 2001.

Y. Freund and 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.

R. Hettich and K.O. Kortanek. Semi-infinite programming: Theory, methods and applications. *SIAM Review*, 3:380–429, September 1993.

G. Lanckriet, N. Cristianini, L. Ghaoui, P. Bartlett, and M. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, 5:27–72, 2004.

R. Meir and G. Rätsch. An introduction to boosting and leveraging. In S. Mendelson and A. Smola, editors, *Advanced Lectures on Machine Learning*, LNCS, pages 119–184. Springer, 2003.

References III

G. Rätsch. *Robust Boosting via Convex Optimization*. PhD thesis, University of Potsdam, Neues Palais 10, 14469 Potsdam, Germany, October 2001.

G. Rätsch and M.K. Warmuth. Efficient margin maximization with boosting. *Journal of Machine Learning Research*, 6:2131–2152, 2005.

G. Rätsch, B. Schölkopf, S. Mika, and K.-R. Müller. SVM and Boosting: One class. Technical Report 119, GMD FIRST, Berlin, November 2000.

G. Rätsch, A. Demiriz, and K. Bennett. Sparse regression ensembles in infinite and finite hypothesis spaces. *Machine Learning*, 48(1-3): 193–221, 2002.

C. Rudin, I. Daubechies, and R.E. Schapire. Dynamics of AdaBoost: Cyclic behavior and convergence of margins. pages 1557–1595, December 2004.

References IV

Rocco A. Servedio. Smooth boosting and learning with malicious noise. *Journal of Machine Learning Research*, 4:633–648, 2003.

S. Sonnenburg, G. Rätsch, C. Schäfer, and B. Schölkopf. Large scale multiple kernel learning. *Journal of Machine Learning Research*, pages 1531–1565, July 2006.

J. von Neumann. Zur Theorie der Gesellschaftsspiele. *Math. Ann.*, 100: 295–320, 1928.

Manfred K Warmuth, Karen Glocer, and Gunnar Rätsch. Boosting algorithms for maximizing the soft margin. In *Advances in Neural Information Processing Systems (NIPS'08)*. MIT Press, 2008. In press.

M.K. Warmuth, J. Liao, and G. Rätsch. Totally corrective boosting algorithms that maximize the margin. In W. Cohen and A. Moore, editors, *Proc. ICML'06*, pages 1001–1008. ACM Press, 2006.