Solving Semi-Infinite Linear Programs using Boosting-like Methods

Gunnar Rätsch

Friedrich Miescher Laboratory of the Max Planck Society, Tübingen, Germany

October 10, 2006

17^{th} International Conference on Algorithmic Learning Theory

1. Some Results on Boosting
   - Basics
   - Large Margins
   - Convergence
   - Illustrative Experiments
2. Semi-Infinite Linear Programming
   - Connections to Boosting
   - Other Methods
3. Applications
   - Ab initio Gene Finding
   - Multiple Kernel Learning
   - Learning Structured Outputs

Protocol of Boosting

- Maintain a distribution $\mathbf{d}^t$ 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, 1995]

Protocol of Boosting

- Maintain a distribution $\mathbf{d}^t$ 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, 1995]

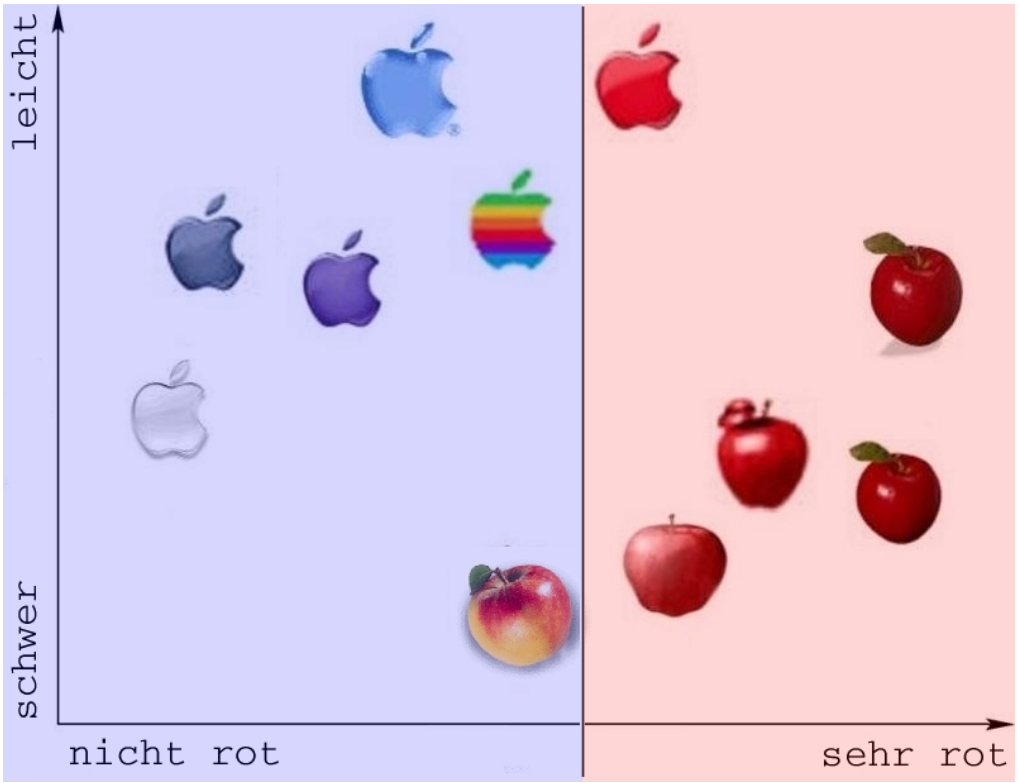
Protocol of Boosting

- Maintain a distribution $\mathbf{d}^t$ 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, 1995]

Boosting: 1st Iteration



First hypothesis:

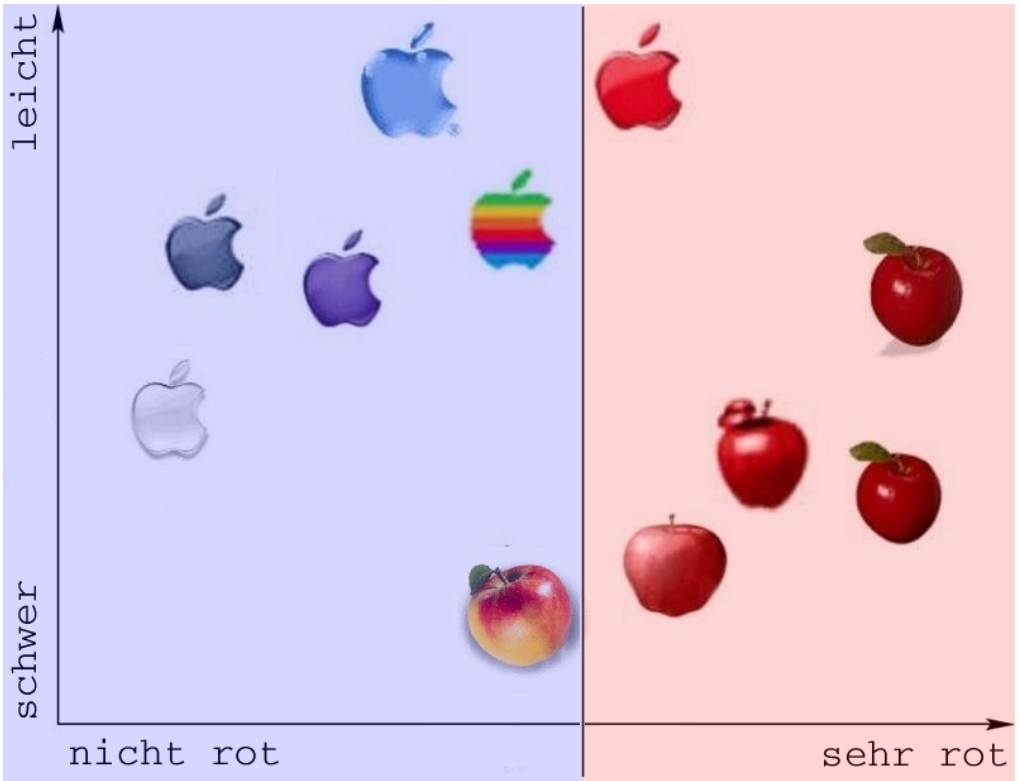
- Error rate: $\frac{2}{11}$

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

- Edge: $\frac{9}{22}$

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

Boosting: 1st Iteration



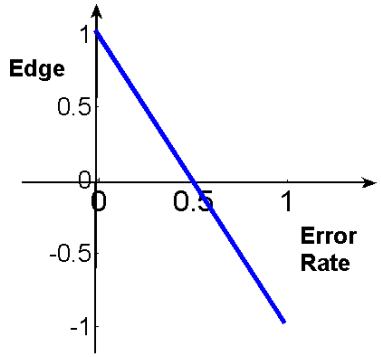
First hypothesis:

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

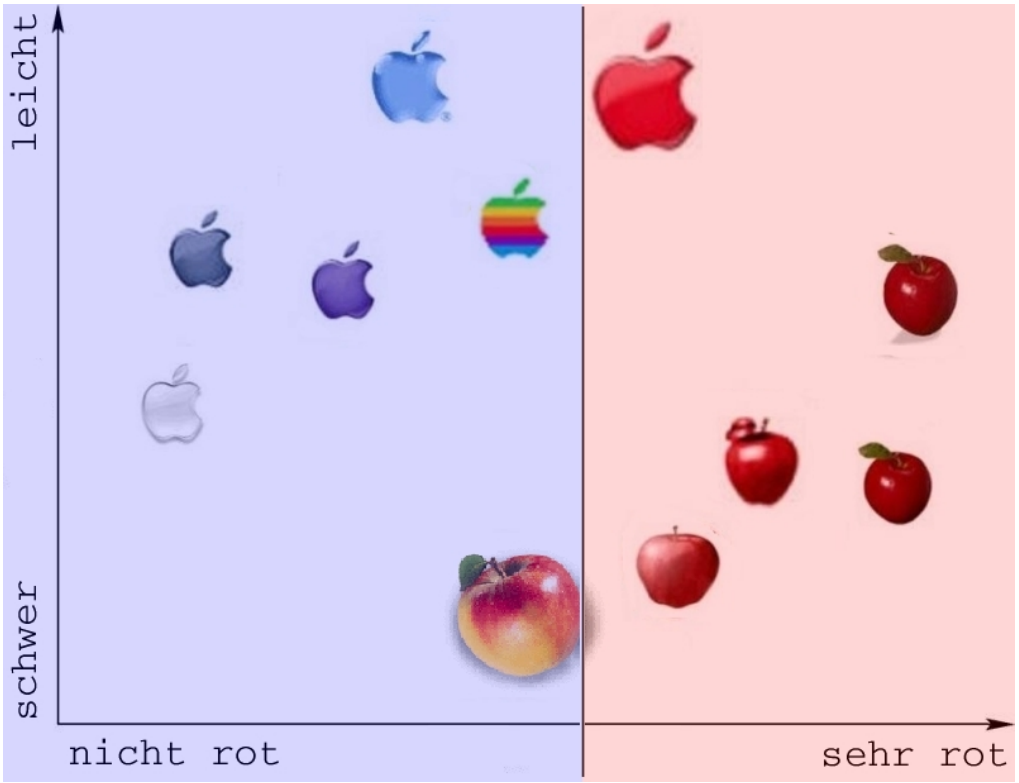
$$\epsilon_t = \sum_{n=1}^{N} d_n^t \mathbf{I}(h_t(\mathbf{x}_n) = y_n)$$

- Edge: $\frac{9}{22}$

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



Update Distribution

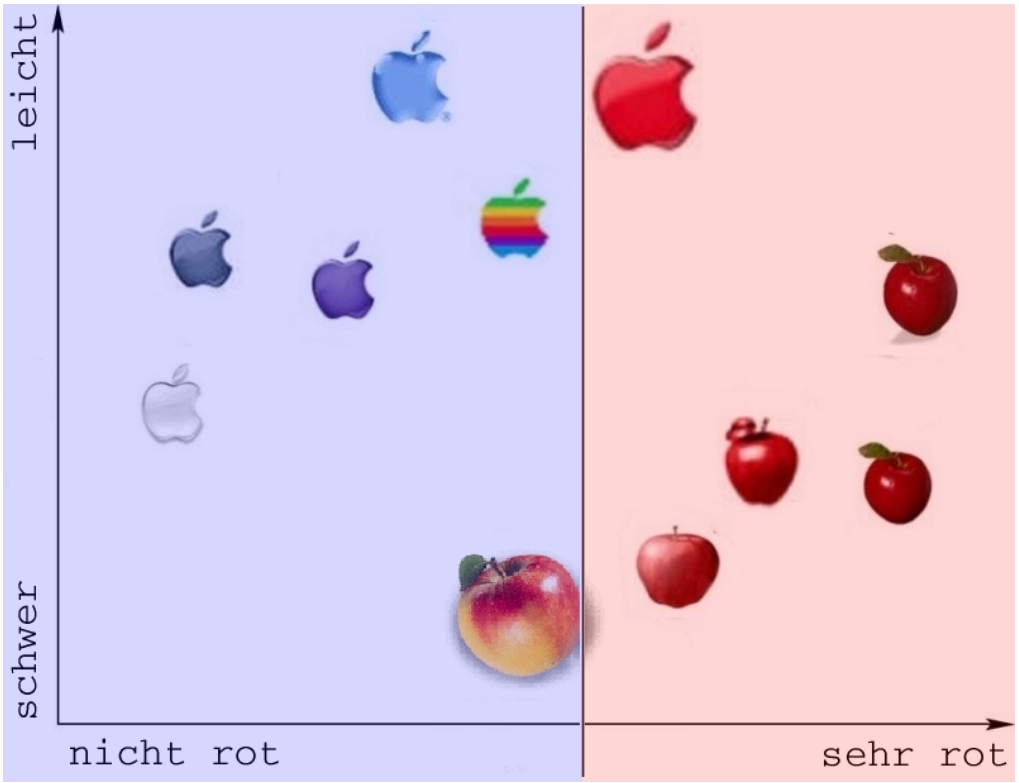


Misclassified examples
$\Rightarrow$ Increased weights

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
$\Rightarrow$ Increased weights

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

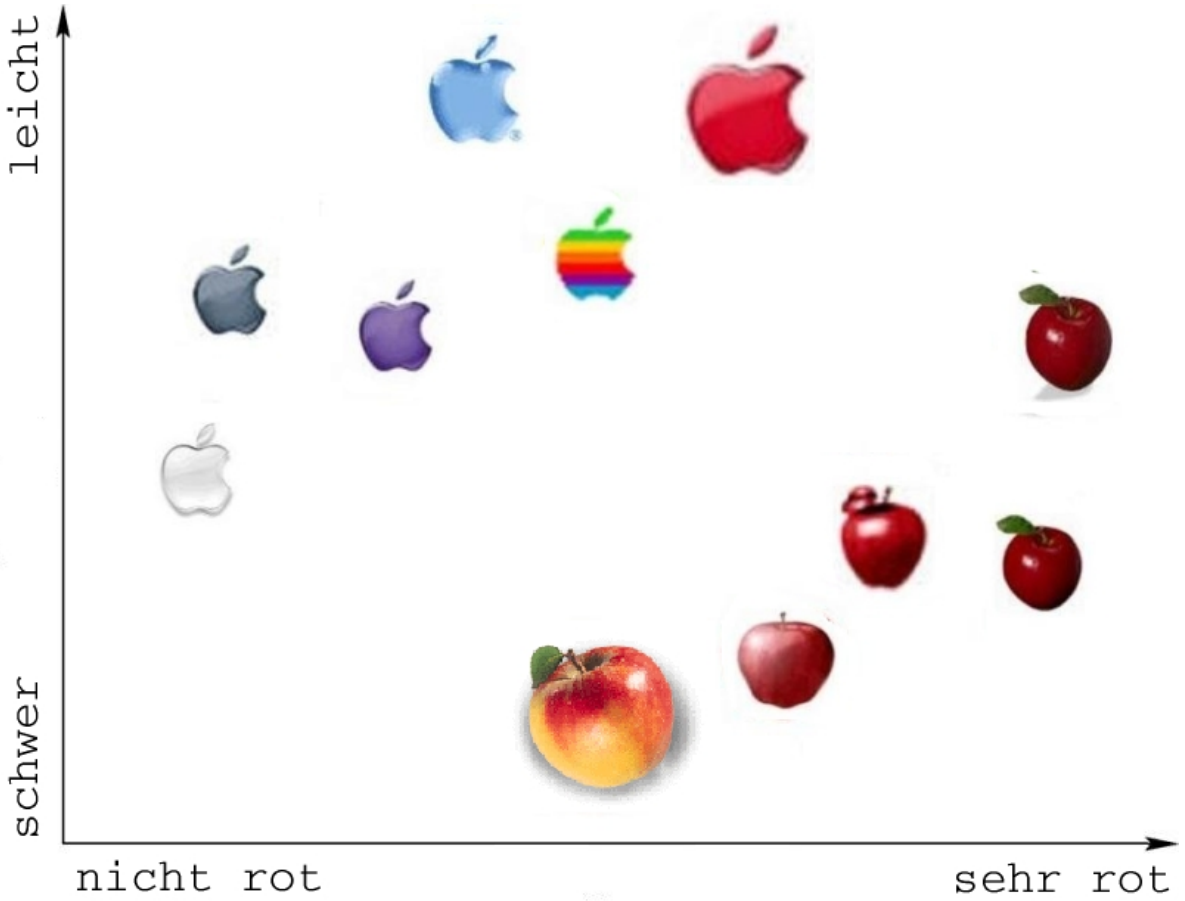
$$\begin{aligned} \min_{\mathbf{d}} \quad & \Delta(\mathbf{d}, \mathbf{d}^t) \\ \text{s.t.} \quad & \sum_{n=1}^{N} d_n y_n h_t(\mathbf{x}_n) = 0 \\ & \mathbf{d} \in \mathcal{P}^N, \end{aligned}$$

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.

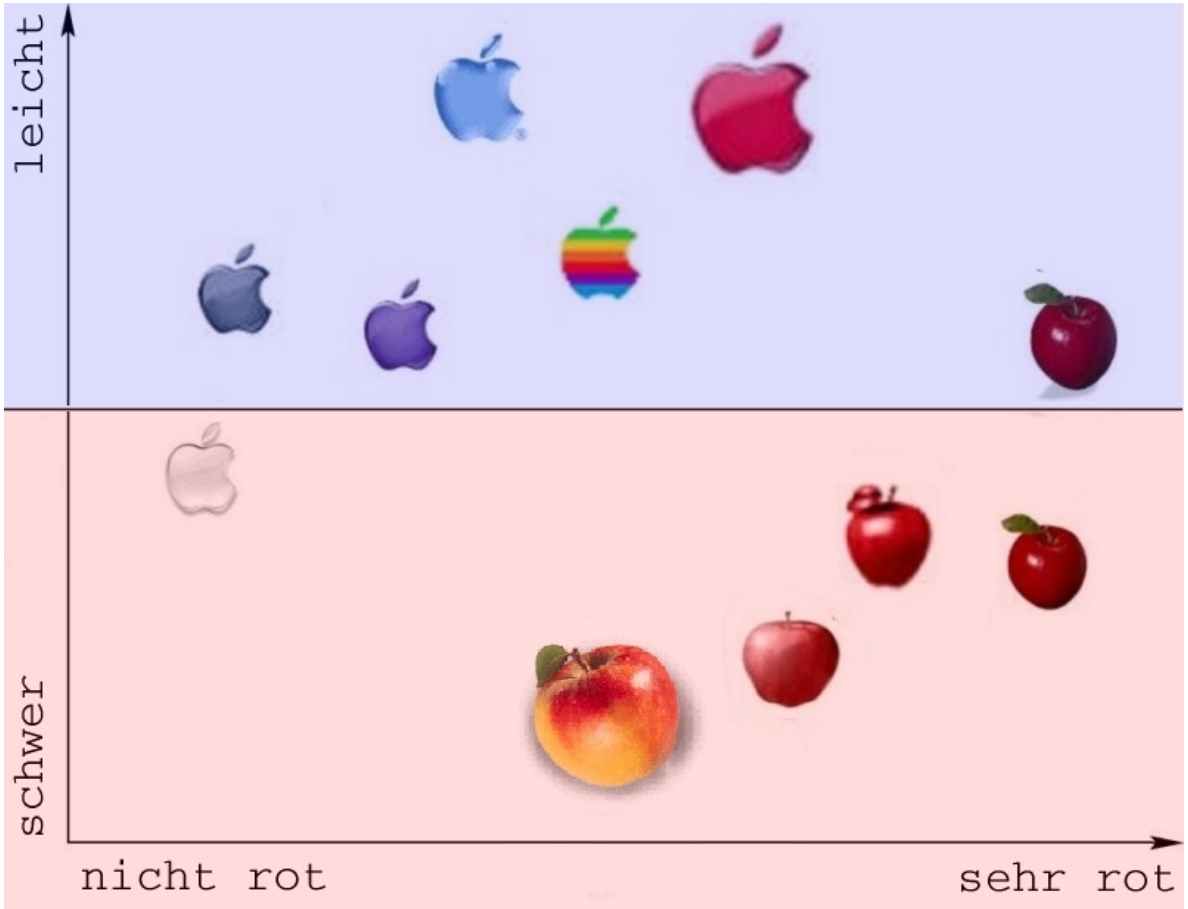
[Lafferty, 1999; Kivinen & Warmuth, 1999]

Before 2nd Iteration

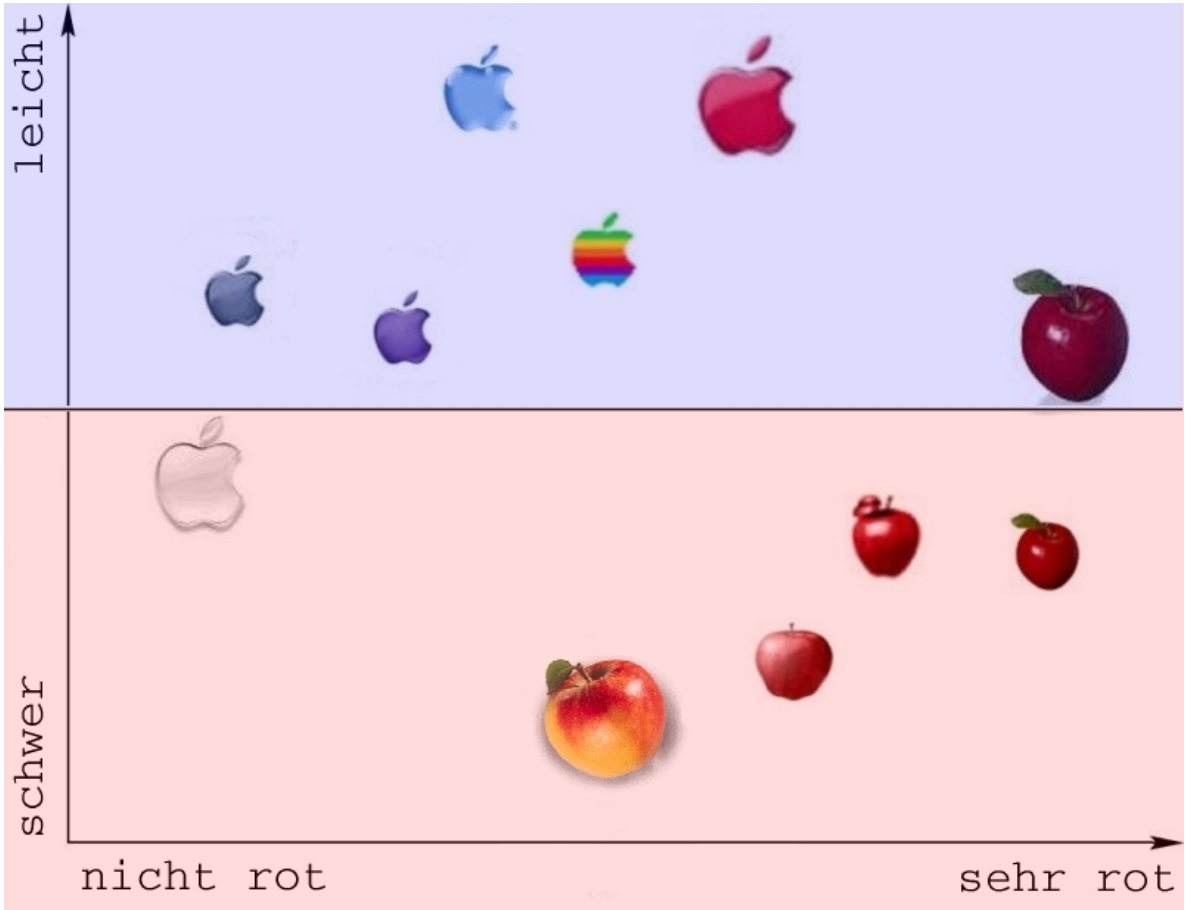


Boosting: 2nd Hypothesis

Edge $\gamma > \nu$



Update Distribution



Edge $\gamma = 0$

AdaBoost update sets edge of last hypothesis to 0

Which constraints?

Corrective: Single constraint

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

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

Totally corrective: One constraint per past weak hypothesis

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

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

[Lafferty, 1999; Kiwinen & Warmuth, 1999]

Which constraints?

Corrective: Single constraint

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

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

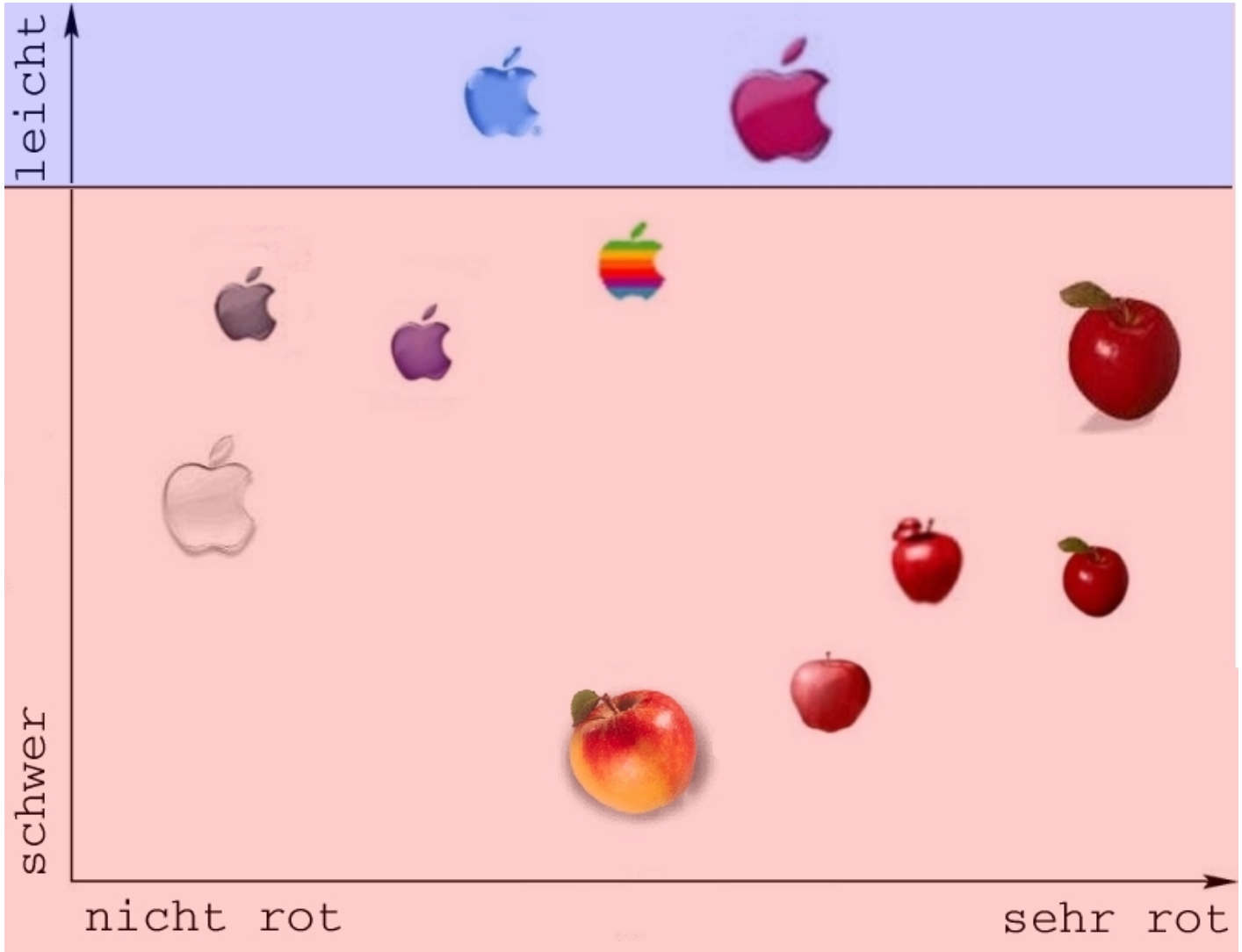
Totally corrective: One constraint per past weak hypothesis

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

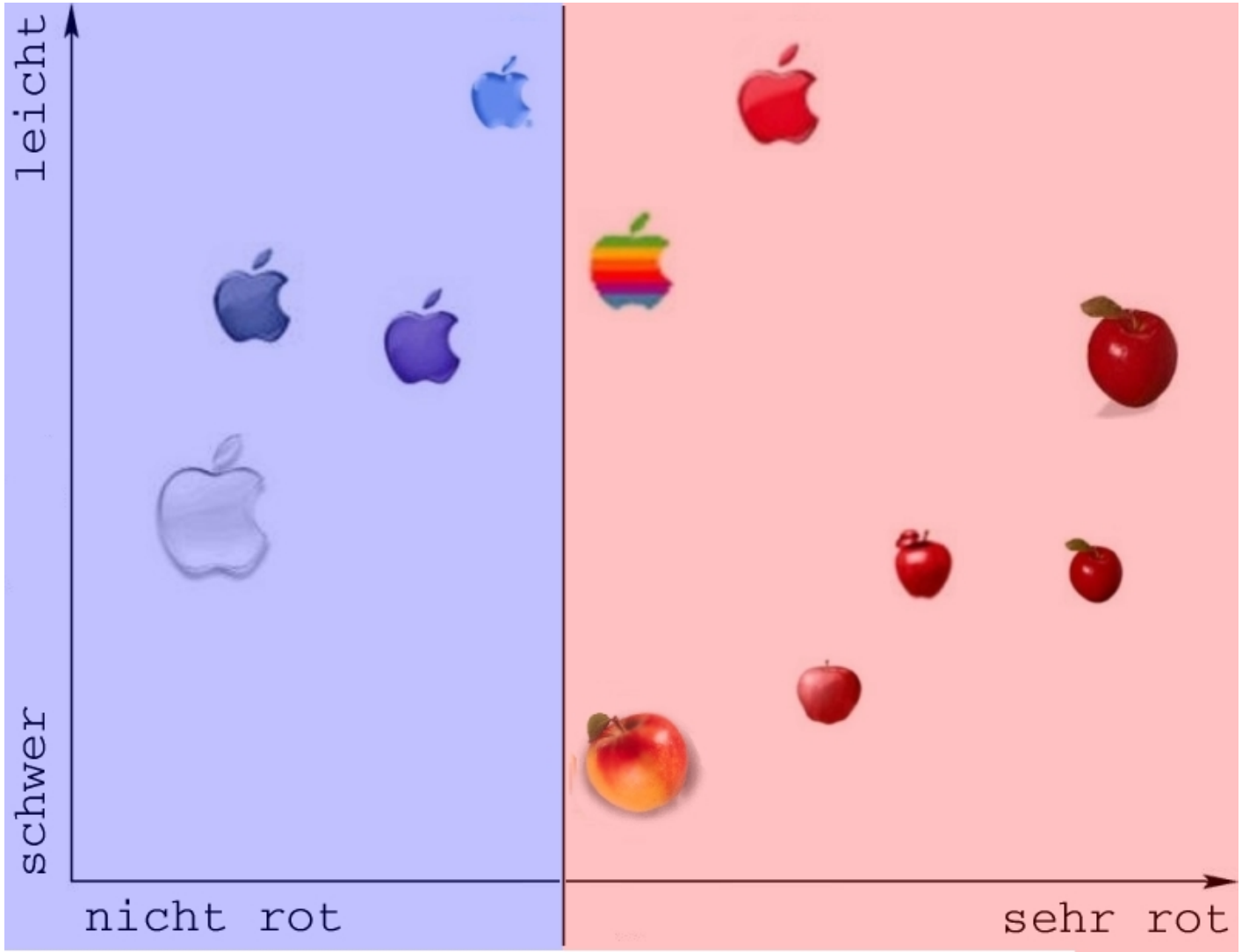
$$\text{s.t.} \quad \sum_{n=1}^{N} d_n y_n h_r(\mathbf{x}_n) = 0 \qquad \text{for } r = 1, \ldots, t$$

[Lafferty, 1999; Kiwinen & Warmuth, 1999]

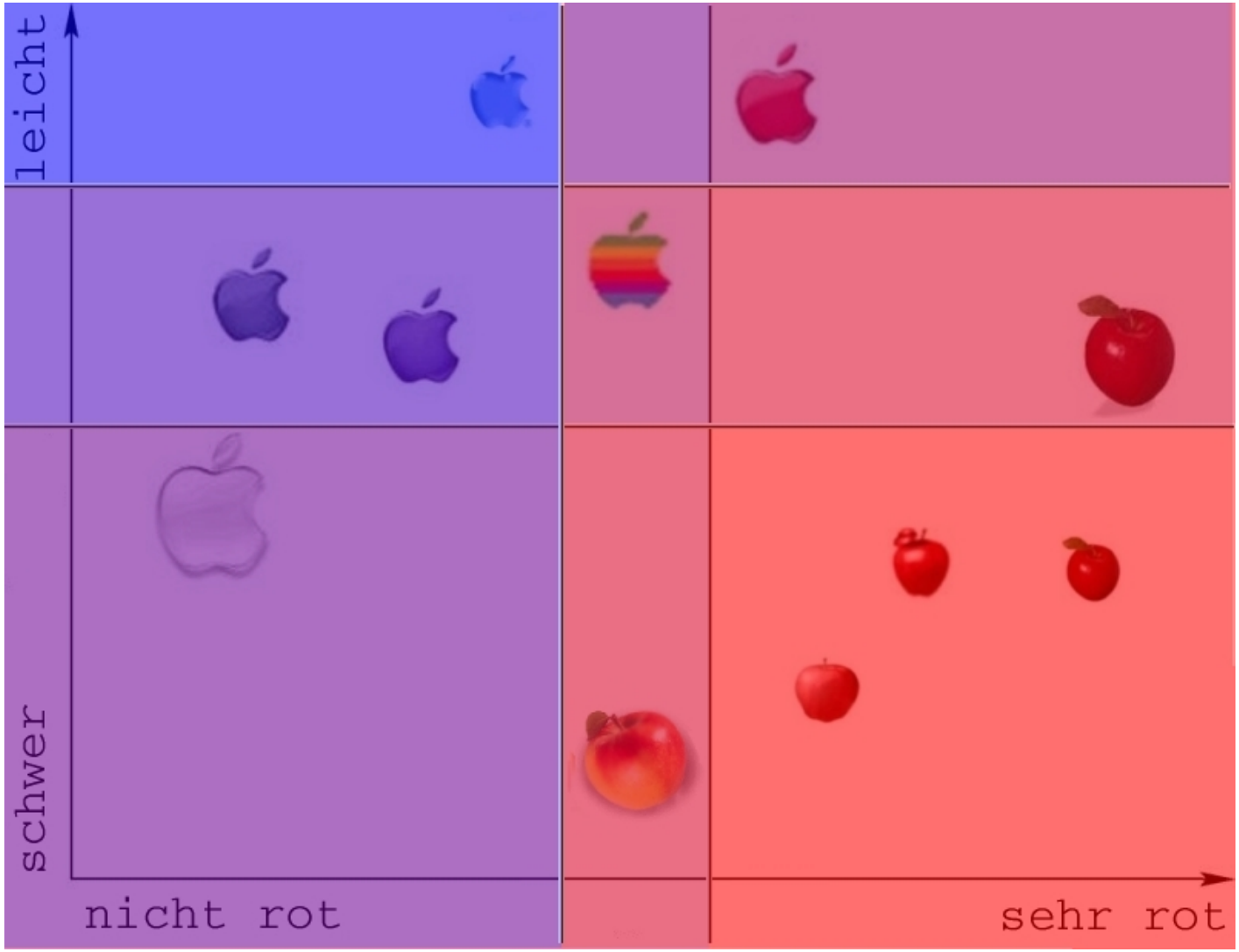
Boosting: 3nd Hypothesis



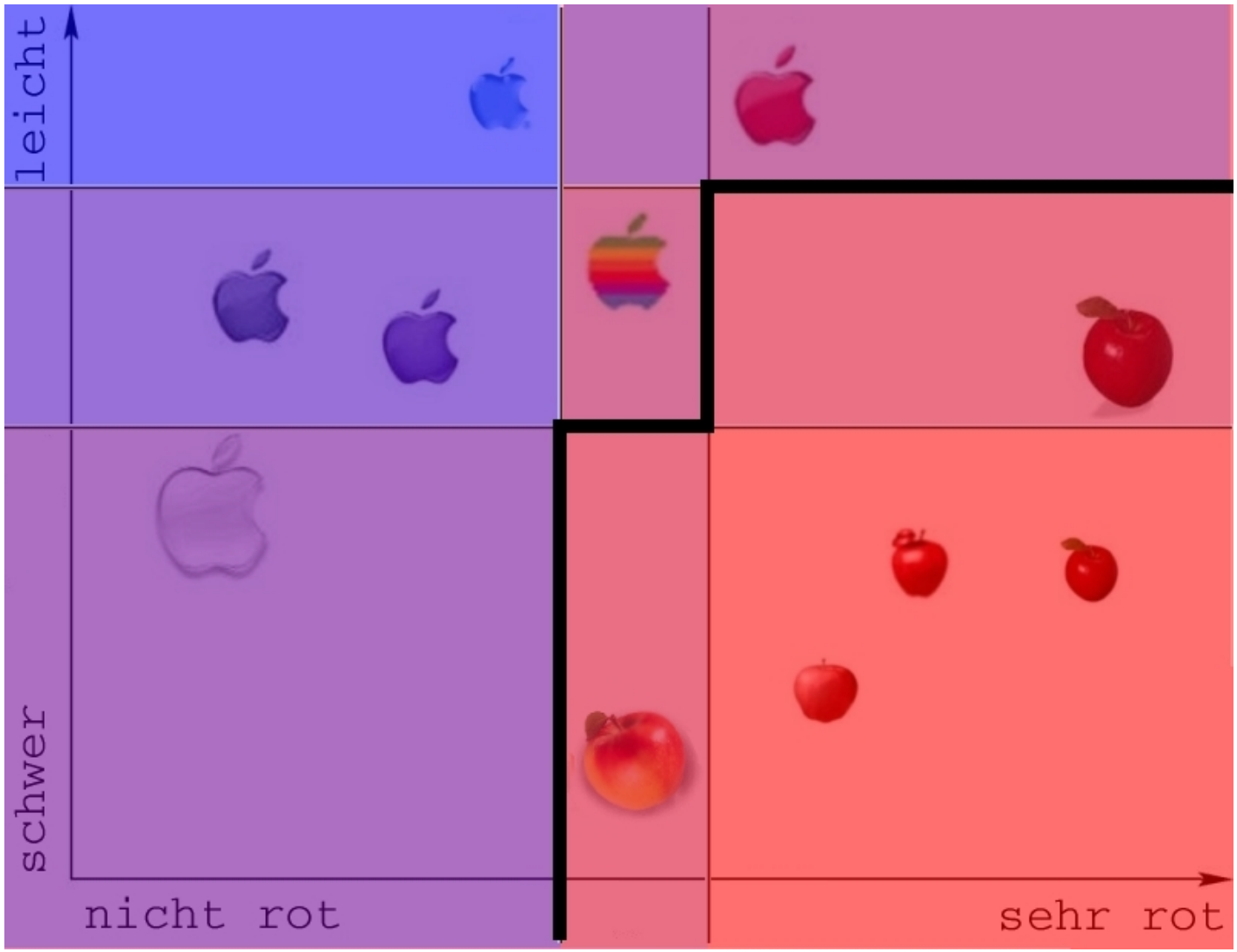
Boosting: 4th Hypothesis



All Hypotheses



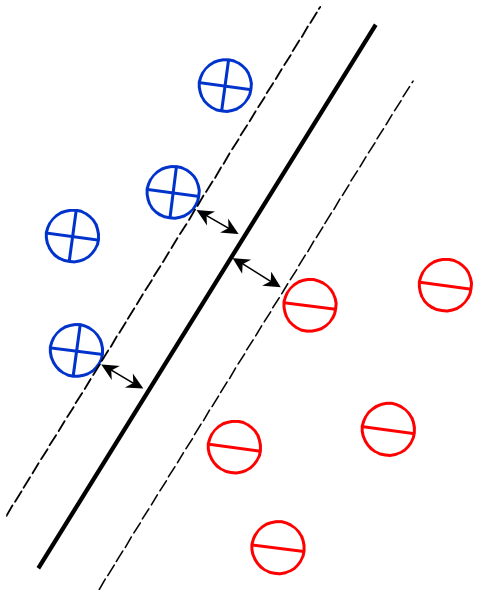
Decision: $f_\alpha(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x}) > 0$?



Large Margins

- Margin of the combined hypothesis $f_{\boldsymbol{\alpha}}$ for example $(\mathbf{x}_n, y_n)$

$$\begin{aligned} \rho_n(\boldsymbol{\alpha}) &= y_n f_{\boldsymbol{\alpha}}(\mathbf{x}_n) \\ &= y_n \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x}_n) \qquad (\boldsymbol{\alpha} \in \mathcal{P}^T) \end{aligned}$$

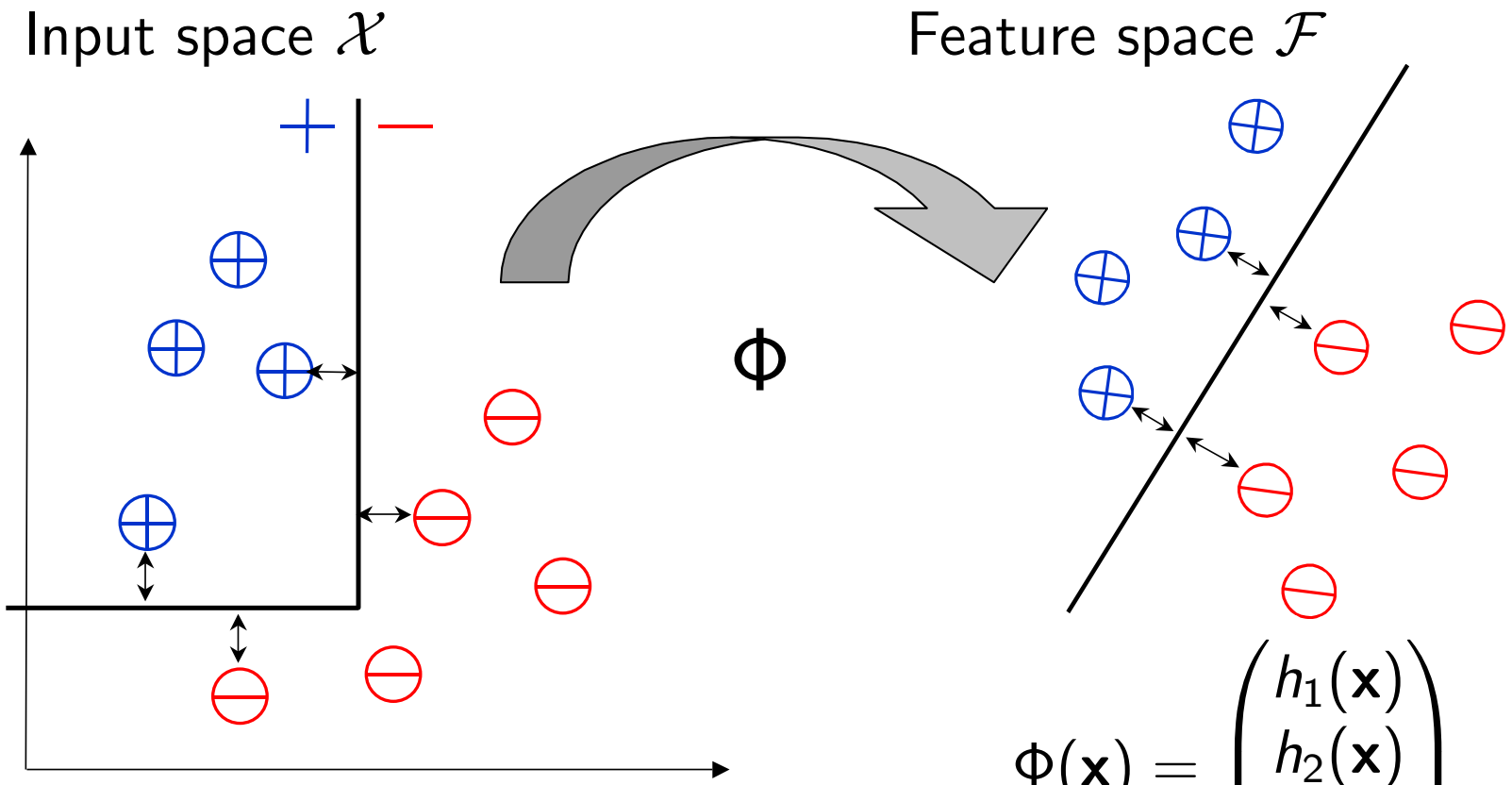


Margin of set of examples is minimum over examples

$$\rho(\boldsymbol{\alpha}) := \min_n \rho_n(\boldsymbol{\alpha})$$

[Freund, Schapire, Bartlett & Lee, 1998]

Large Margin and Linear Separation



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

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

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

[Mangasarian, 1999; G.R., Mika, Schölkopf & Müller, 2002]

Edge vs. Margin

Edge

- Measurement of "goodness" of a hypothesis w.r.t. a distribution
- Edge of a hypothesis h for a distribution $\mathbf{d}$ on the examples

$$\gamma_h(\mathbf{d}) = \sum_{n=1}^{N} d_n \; y_n h(\mathbf{x}_n) \qquad \mathbf{d} \in \mathcal{P}^N$$

Margin

- Measure for "confidence" in prediction for a hypothesis weighting
- Margin of example n for current hypothesis weighting α

$$\rho_n(\alpha) \;=\; y_n f_\alpha(\mathbf{x}_n) = y_n \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x}_n) \qquad (\alpha \in \mathcal{P}^T)$$

What is the connection?

[Breiman, 1999]

Edge vs. Margin

Edge

- Measurement of "goodness" of a hypothesis w.r.t. a distribution
- Edge of a hypothesis h for a distribution $\mathbf{d}$ on the examples

$$\gamma_h(\mathbf{d}) = \sum_{n=1}^{N} d_n \; y_n h(\mathbf{x}_n) \qquad \mathbf{d} \in \mathcal{P}^N$$

Margin

- Measure for "confidence" in prediction for a hypothesis weighting
- Margin of example n for current hypothesis weighting $\boldsymbol{\alpha}$

$$\rho_n(\boldsymbol{\alpha}) \;=\; y_n f_{\boldsymbol{\alpha}}(\mathbf{x}_n) = y_n \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x}_n) \qquad (\boldsymbol{\alpha} \in \mathcal{P}^T)$$

What is the connection?

[Breiman, 1999]

Edge vs. Margin

Edge

- Measurement of "goodness" of a hypothesis w.r.t. a distribution
- Edge of a hypothesis h for a distribution $\mathbf{d}$ on the examples

$$\gamma_h(\mathbf{d}) = \sum_{n=1}^{N} d_n \; y_n h(\mathbf{x}_n) \qquad \mathbf{d} \in \mathcal{P}^N$$

Margin

- Measure for "confidence" in prediction for a hypothesis weighting
- Margin of example n for current hypothesis weighting $\boldsymbol{\alpha}$

$$\rho_n(\boldsymbol{\alpha}) \;=\; y_n f_{\boldsymbol{\alpha}}(\mathbf{x}_n) = y_n \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x}_n) \qquad (\boldsymbol{\alpha} \in \mathcal{P}^T)$$

What is the connection? [Breiman, 1999]

von Neumann's Minimax-Theorem

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

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

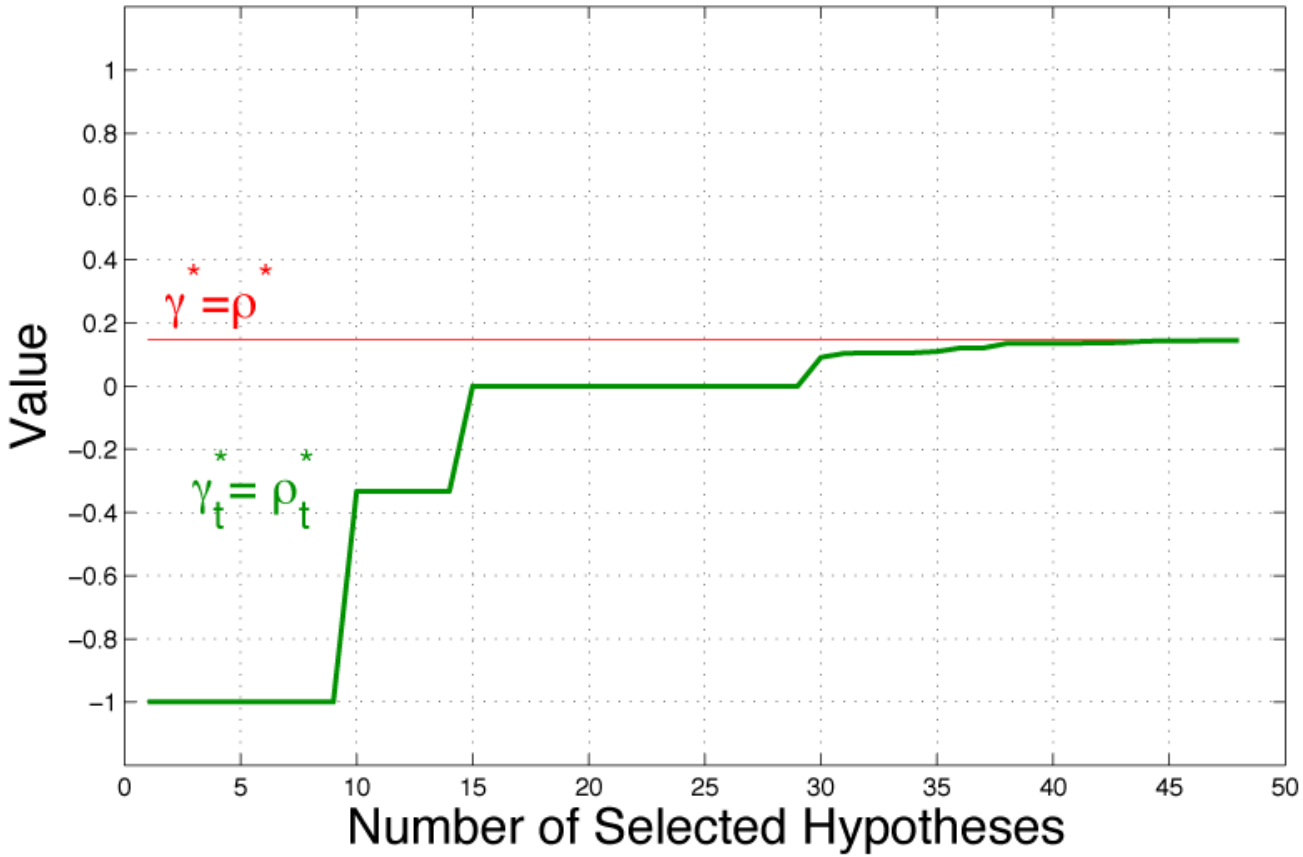
$$\text{maximum margin: } \rho_t^* = \max_{\boldsymbol{\alpha} \in \mathcal{P}^t} \underbrace{\min_n \overbrace{y_n f_{\boldsymbol{\alpha}}(\mathbf{x}_n)}^{\text{margin of } (\mathbf{x}_n, y_n)}}_{\text{margin of S}}$$

$$\text{Duality: } \gamma_t^* = \rho_t^*$$

[von Neumann, 1928]

Value non-decreasing

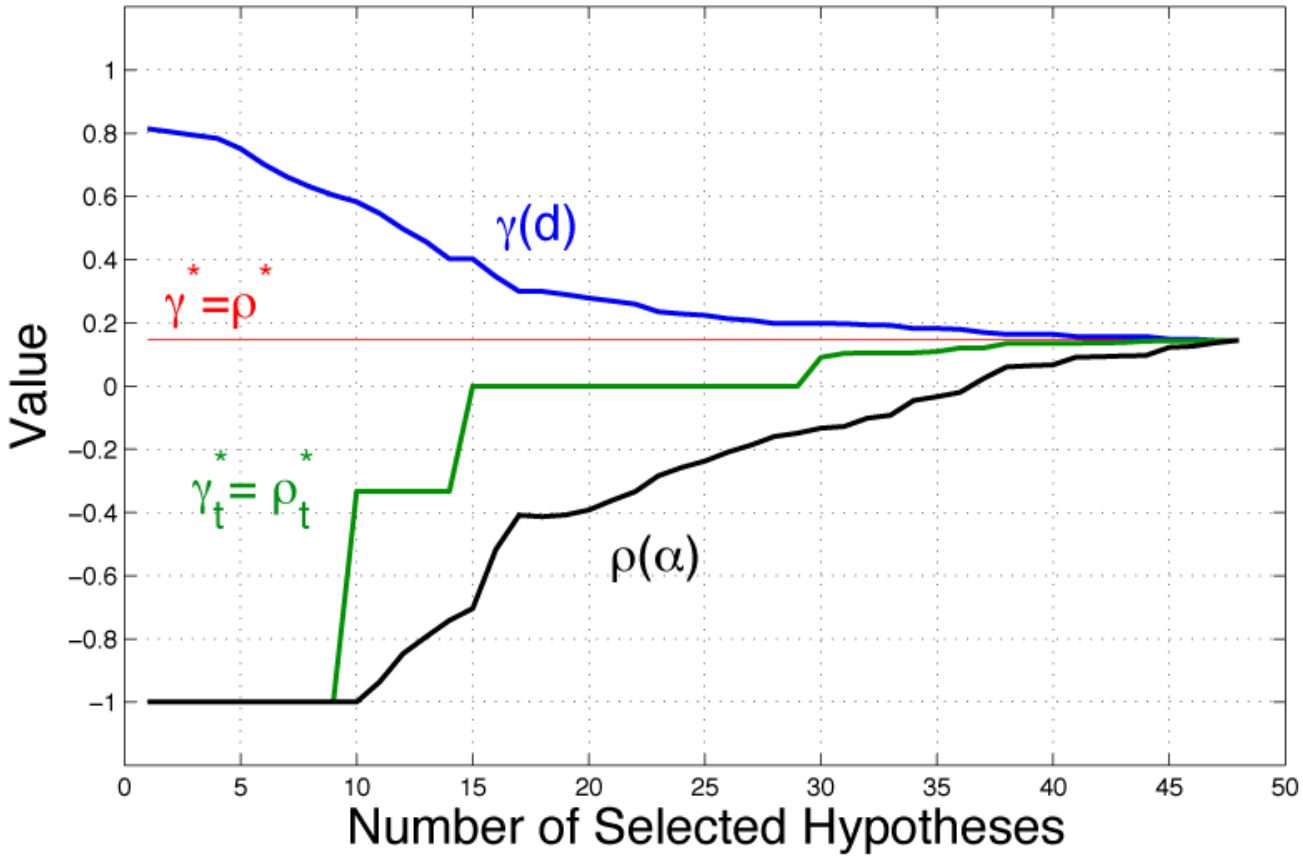
γ^*, ρ^*: edge/margin for all hypotheses



Duality gap

For any non-optimal $\mathbf{d} \in \mathcal{P}^N$ and $\boldsymbol{\alpha} \in \mathcal{P}^t$,

$$\gamma(\mathbf{d}) \geq \gamma_t^* = \rho_t^* \geq \rho(\boldsymbol{\alpha})$$



How Large is the Maximal Margin?



Assumptions on Weak learner

For any distribution $\mathbf{d}$ on the examples, the weak learner returns a hypothesis h with edge $\gamma_h(\mathbf{d})$ at least g.
Best case: $g = \rho^* = \gamma^*$.

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

How Large is the Maximal Margin?



Assumptions on Weak learner

For any distribution $\mathbf{d}$ on the examples, the weak learner returns a hypothesis h with edge $\gamma_h(\mathbf{d})$ at least g.
Best case: $g = \rho^* = \gamma^*$.

Implication from Minimax Theorem

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

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

How Large is the Maximal Margin?

MAX-PLANCK-GESELLSCHAFT

Assumptions on Weak learner

For any distribution $\mathbf{d}$ on the examples, the weak learner returns a hypothesis h with edge $\gamma_h(\mathbf{d})$ at least g.
Best case: $g = \rho^* = \gamma^*$.

Implication from Minimax Theorem

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

Idea to iteratively solve LP: LPBoost

Add "best" hypothesis $h = \operatorname{argmax} \gamma_h(\mathbf{d}^t)$ to $\mathcal{H}^{t+1}$ and resolve

$$\mathbf{d}^{t+1} = \operatorname*{argmin}_{d \in \mathcal{P}^N} \max_{h \in \mathcal{H}^{t+1}} \gamma_h(\mathbf{d}).$$

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

How Large is the Maximal Margin?



Assumptions on Weak learner

For any distribution $\mathbf{d}$ on the examples, the weak learner returns a hypothesis h with edge $\gamma_h(\mathbf{d})$ at least g.
Best case: $g = \rho^* = \gamma^*$.

What about AdaBoost?

- May "oscillate"
- Does not find maximizing $\boldsymbol{\alpha}$ (counter examples)
- But there some guarantees:
  - $\rho(\boldsymbol{\alpha}^t) \geq 0$ after $2 \ln N / g^2$ iterations
  - $\rho(\boldsymbol{\alpha}^t) \geq g/2$ in the limit

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

How to Maximize the Margin?



Modify AdaBoost for maximizing margin

- Arc-GV asymptotically maximizes the margin
  - quite slow, no converge rates
- LPBoost uses a Linear Programming solver
  - Often very fast in practice, but no converge rates
- AdaBoost* requires $\frac{2\log(N)}{\nu^2}$ iterations to get $\rho^t \in [\rho^* - \nu, \rho^*]$
  - Slow in practice, i.e. not faster than theory predicts
- TotalBoost requires $\frac{2\log(N)}{\nu^2}$ iterations to get $\rho^t \in [\rho^* - \nu, \rho^*]$
  - Fast in practice
  - Combination of benefits

[G.R. & Warmuth, 2004; Warmuth et al., 2006]

How to Maximize the Margin?



Modify AdaBoost for maximizing margin

- Arc-GV asymptotically maximizes the margin
  - quite slow, no converge rates
- LPBoost uses a Linear Programming solver
  - Often very fast in practice, but no converge rates
- AdaBoost* requires $\frac{2\log(N)}{\nu^2}$ iterations to get $\rho^t \in [\rho^* - \nu, \rho^*]$
  - Slow in practice, i.e. not faster than theory predicts
- TotalBoost requires $\frac{2\log(N)}{\nu^2}$ iterations to get $\rho^t \in [\rho^* - \nu, \rho^*]$
  - Fast in practice
  - Combination of benefits

[G.R. & Warmuth, 2004; Warmuth et al., 2006]

How to Maximize the Margin?



Modify AdaBoost for maximizing margin

- Arc-GV asymptotically maximizes the margin
  - quite slow, no converge rates
- LPBoost uses a Linear Programming solver
  - Often very fast in practice, but no converge rates
- AdaBoost* requires $\frac{2\log(N)}{\nu^2}$ iterations to get $\rho^t \in [\rho^* - \nu, \rho^*]$
  - Slow in practice, i.e. not faster than theory predicts
- TotalBoost requires $\frac{2\log(N)}{\nu^2}$ iterations to get $\rho^t \in [\rho^* - \nu, \rho^*]$
  - Fast in practice
  - Combination of benefits

[G.R. & Warmuth, 2004; Warmuth et al., 2006]

How to Maximize the Margin?

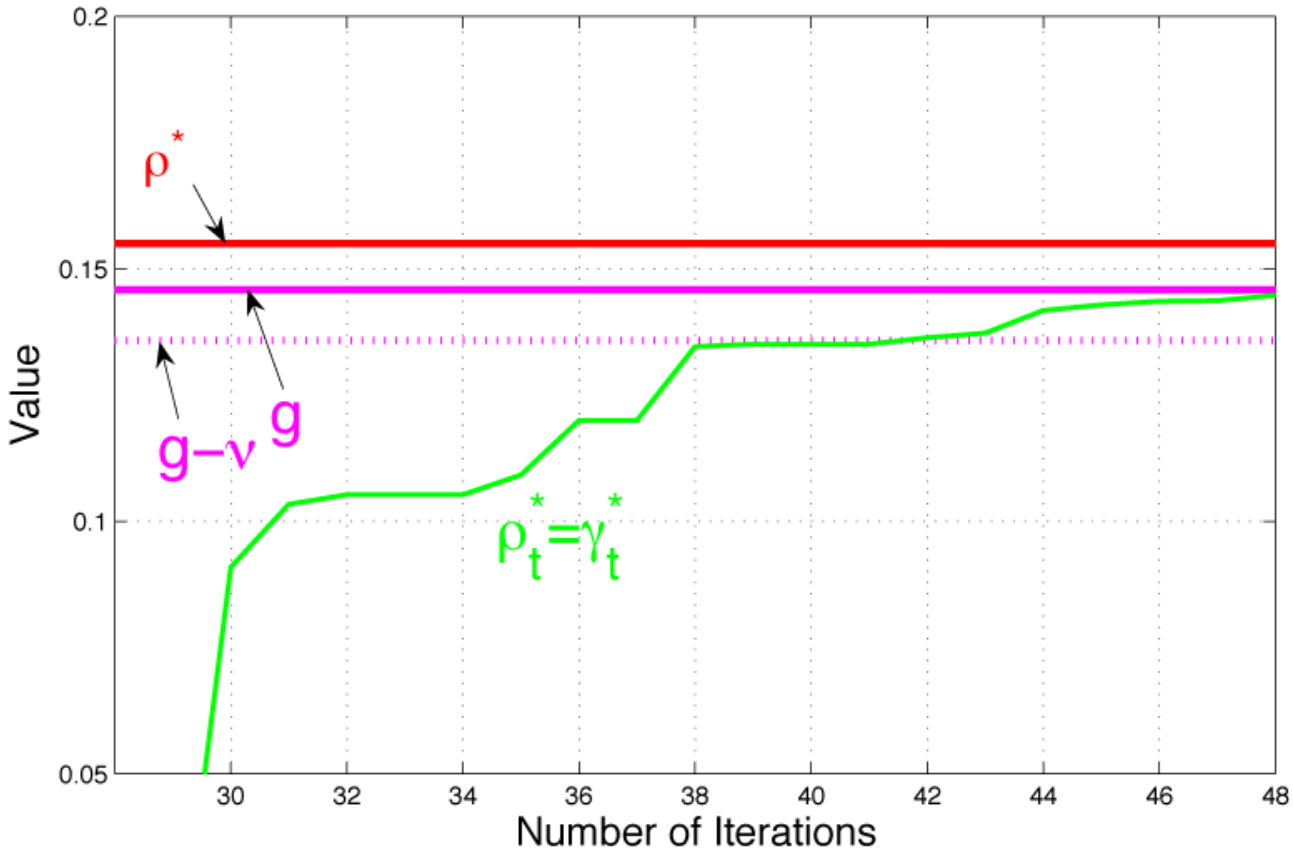


Modify AdaBoost for maximizing margin

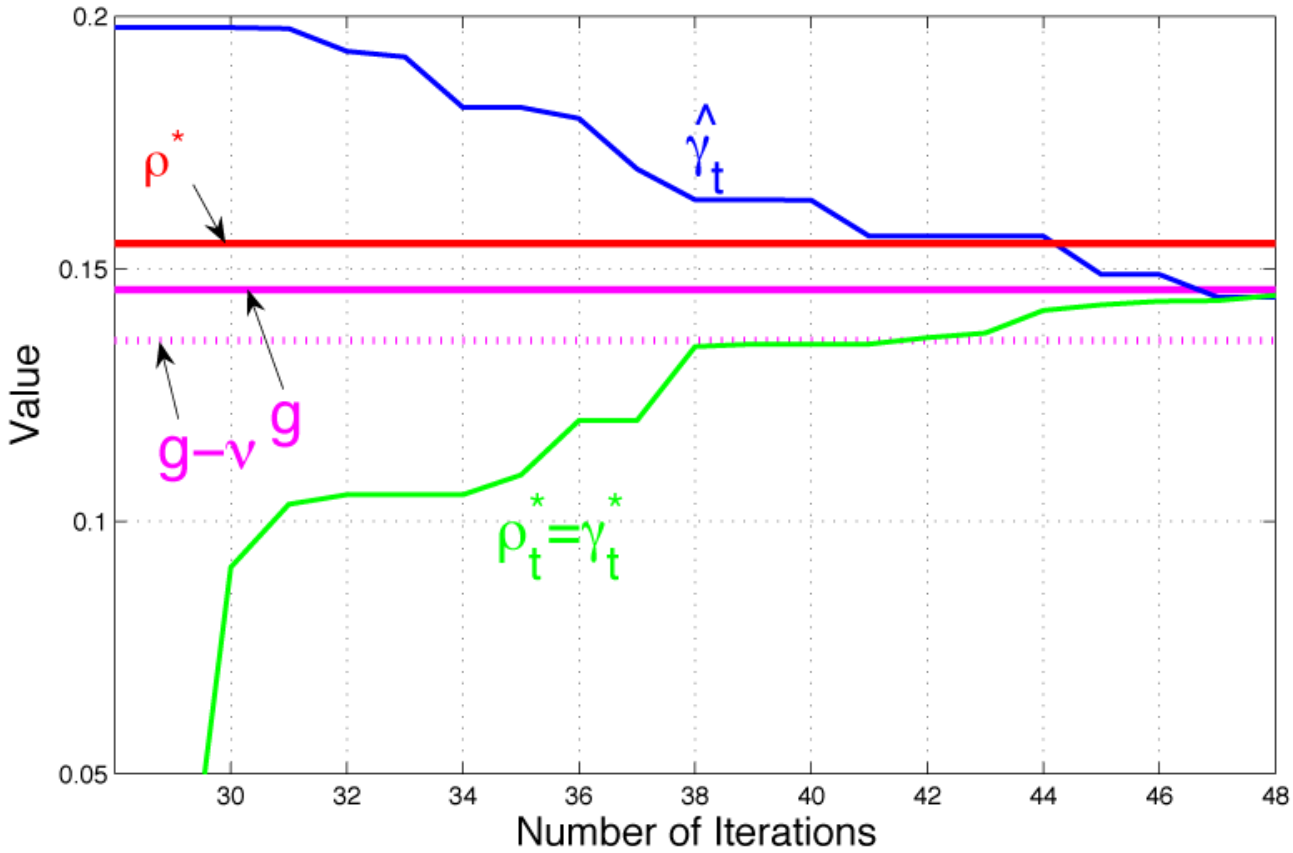
- Arc-GV asymptotically maximizes the margin
  - quite slow, no converge rates
- LPBoost uses a Linear Programming solver
  - Often very fast in practice, but no converge rates
- AdaBoost* requires $\frac{2\log(N)}{\nu^2}$ iterations to get $\rho^t \in [\rho^* - \nu, \rho^*]$
  - Slow in practice, i.e. not faster than theory predicts
- TotalBoost requires $\frac{2\log(N)}{\nu^2}$ iterations to get $\rho^t \in [\rho^* - \nu, \rho^*]$
  - Fast in practice
  - Combination of benefits

[G.R. & Warmuth, 2004; Warmuth et al., 2006]

Want margin $\geq g - \nu$



Want margin $\geq g - \nu$



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



Corrective: Single constraint

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

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

Totally corrective: One constraint per past weak hypothesis

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

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

TotalBoost$_\nu$

1. **Input:** $S = \langle (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N) \rangle$, desired accuracy ν
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 $\widehat{\gamma}_t = (\min_{q=1,\ldots,t} \gamma_q) - \nu$ and solve
      $$\mathbf{d}^{t+1} = \operatorname*{argmin}_{\{\mathbf{d} \in \mathcal{P}^N \mid \mathbf{d} \cdot \mathbf{u}^q \leq \widehat{\gamma}_t, \text{ for } 1 \leq q \leq t\} = \mathcal{C}_t} \Delta(\mathbf{d}, \mathbf{d}^1)$$
   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$_\nu$

1. **Input:** $S = \langle (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N) \rangle$, desired accuracy ν
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 $\widehat{\gamma}_t = (\min_{q=1,\ldots,t} \gamma_q) - \nu$ and solve
      $$\mathbf{d}^{t+1} = \operatorname*{argmin}_{\{\mathbf{d} \in \mathcal{P}^N \mid \mathbf{d} \cdot \mathbf{u}^q \leq \widehat{\gamma}_t, \text{ for } 1 \leq q \leq t\} = \mathcal{C}_t} \Delta(\mathbf{d}, \mathbf{d}^1)$$
   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$_\nu$

1. **Input:** $S = \langle (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N) \rangle$, desired accuracy ν
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 $\widehat{\gamma}_t = (\min_{q=1,\ldots,t} \gamma_q) - \nu$ and solve
   4. **Optimization Problem**

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

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

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 $\{h_1, \ldots, h_T\}$.

Iteration Bound $\lceil \frac{2 \ln N}{\nu^2} \rceil$



Theorem

For $\mathbf{d}^t, \mathbf{d}^{t+1} \in \mathcal{P}^N$ *and* $\mathbf{u} \in [-1,1]^N$,
if $\Delta(\mathbf{d}^{t+1}, \mathbf{d}^t)$ *finite and* $\mathbf{d}^{t+1} \cdot \mathbf{u} \neq \mathbf{d}^t \cdot \mathbf{u}$ *then*

$$\Delta(\mathbf{d}^{t+1}, \mathbf{d}^t) > \frac{(\mathbf{d}^{t+1} \cdot \mathbf{u} - \mathbf{d}^t \cdot \mathbf{u})^2}{2}$$

- $\mathbf{d}^t \cdot \mathbf{u}^t = \gamma_t$
- $\mathbf{d}^{t+1} \cdot \mathbf{u}^t \leq \widehat{\gamma}_t \leq \gamma_t - \nu$
- Thus $\mathbf{d}^t \cdot \mathbf{u}^t - \mathbf{d}^{t+1} \cdot \mathbf{u}^t \geq \nu$ and $\Delta(\mathbf{d}^{t+1}, \mathbf{d}^t) > \frac{\nu^2}{2}$

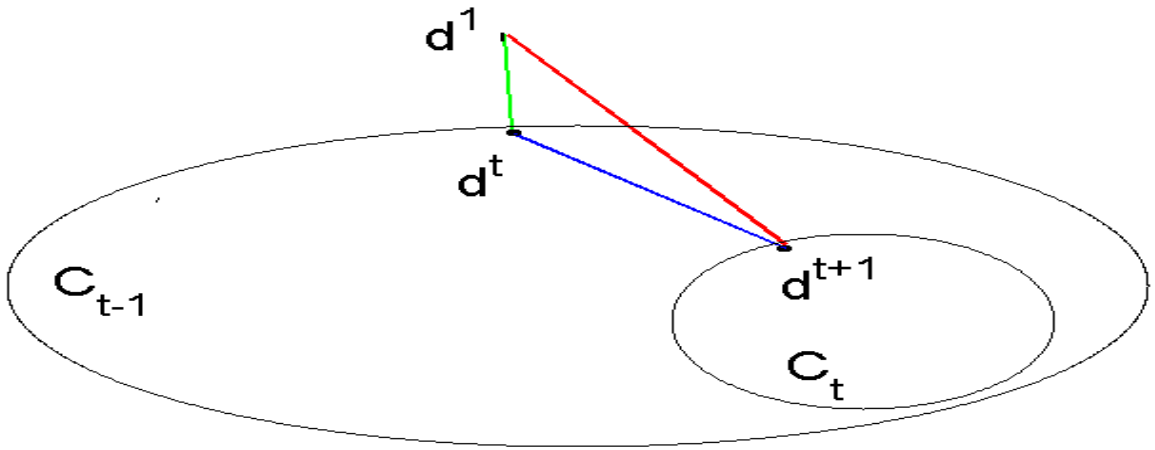
[Warmuth, Liao & G.R., 2006]

Generalized Pythagorean Theorem

$\mathcal{C}_t = \{\mathbf{d} \in \mathcal{P}^N | \mathbf{d} \cdot \mathbf{u}^q \leq \widehat{\gamma}_t, 1 \leq q \leq t\}, \mathcal{C}_0 = \mathcal{P}^N, \mathcal{C}_t \subseteq \mathcal{C}_{t-1}$
$\mathbf{d}^t$ is projection of $\mathbf{d}^1$ onto $\mathcal{C}_{t-1}$ at iteration $t-1$

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

$$\Delta(\mathbf{d}^{t+1}, \mathbf{d}^1) \geq \Delta(\mathbf{d}^t, \mathbf{d}^1) + \Delta(\mathbf{d}^{t+1}, \mathbf{d}^t)$$



[Herbster & Warmuth, 2001]

Sketch of Proof

$$
\begin{array}{rccccc}
1: & \Delta(\mathbf{d}^2,\mathbf{d}^1) - \Delta(\mathbf{d}^1,\mathbf{d}^1) & \geq & \Delta(\mathbf{d}^2,\mathbf{d}^1) & > & \frac{\nu^2}{2} \\
2: & \Delta(\mathbf{d}^3,\mathbf{d}^2) - \Delta(\mathbf{d}^2,\mathbf{d}^1) & \geq & \Delta(\mathbf{d}^3,\mathbf{d}^2) & > & \frac{\nu^2}{2} \\
 & \dots & & \dots & & \\
t: & \Delta(\mathbf{d}^{t+1},\mathbf{d}^1) - \Delta(\mathbf{d}^t,\mathbf{d}^1) & \geq & \Delta(\mathbf{d}^{t+1},\mathbf{d}^t) & > & \frac{\nu^2}{2} \\
 & \dots & & \dots & & \\
T-1: & \Delta(\mathbf{d}^T,\mathbf{d}^1) - \Delta(\mathbf{d}^{T-1},\mathbf{d}^1) & \geq & \Delta(\mathbf{d}^T,\mathbf{d}^{T-1}) & > & \frac{\nu^2}{2}
\end{array}
$$

Therefore, $T \leq \lceil \frac{2 \ln N}{\nu^2} \rceil$

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

Illustrative Experiments

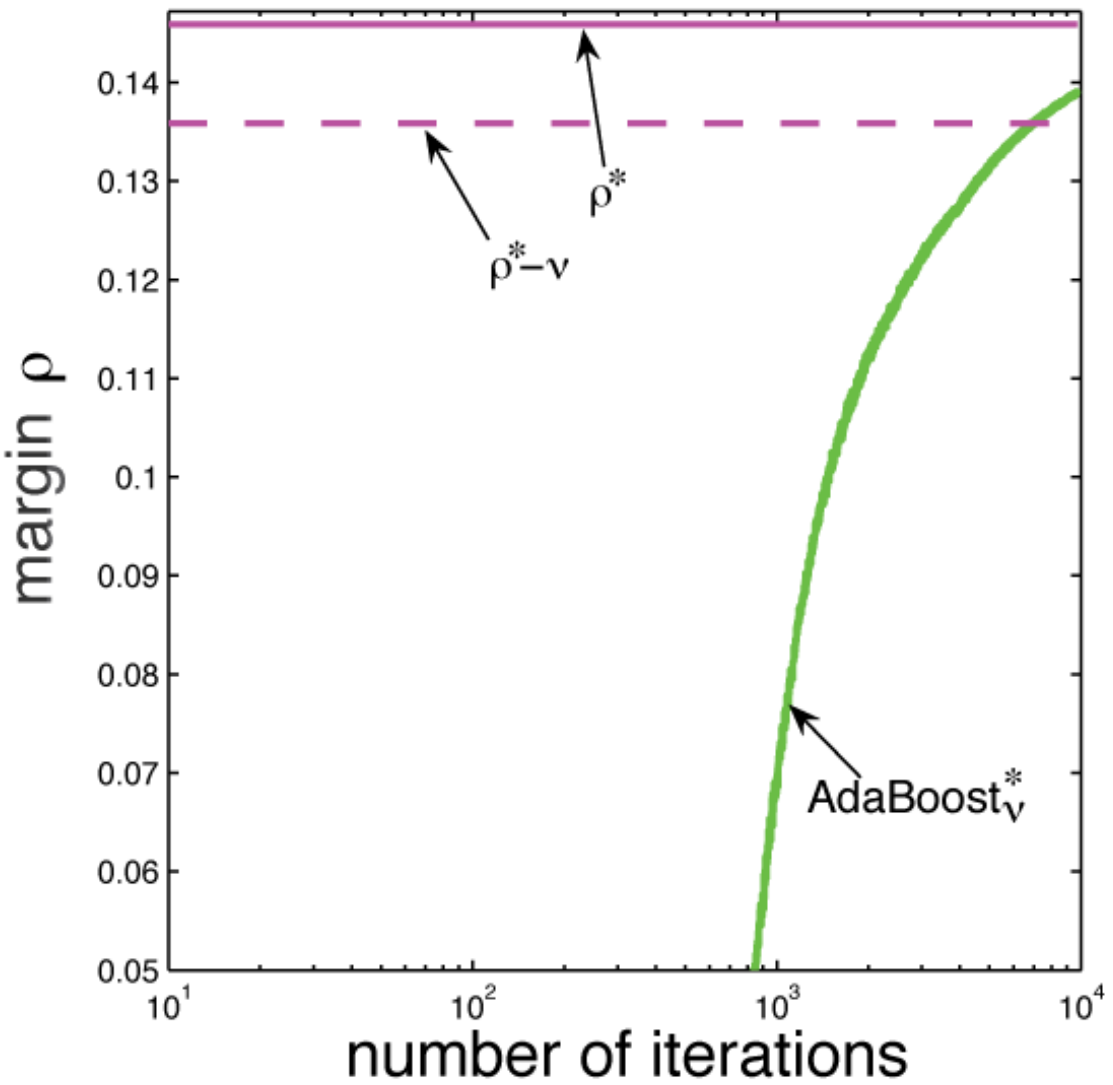


Cox-1 dataset from Telik Inc.

- 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

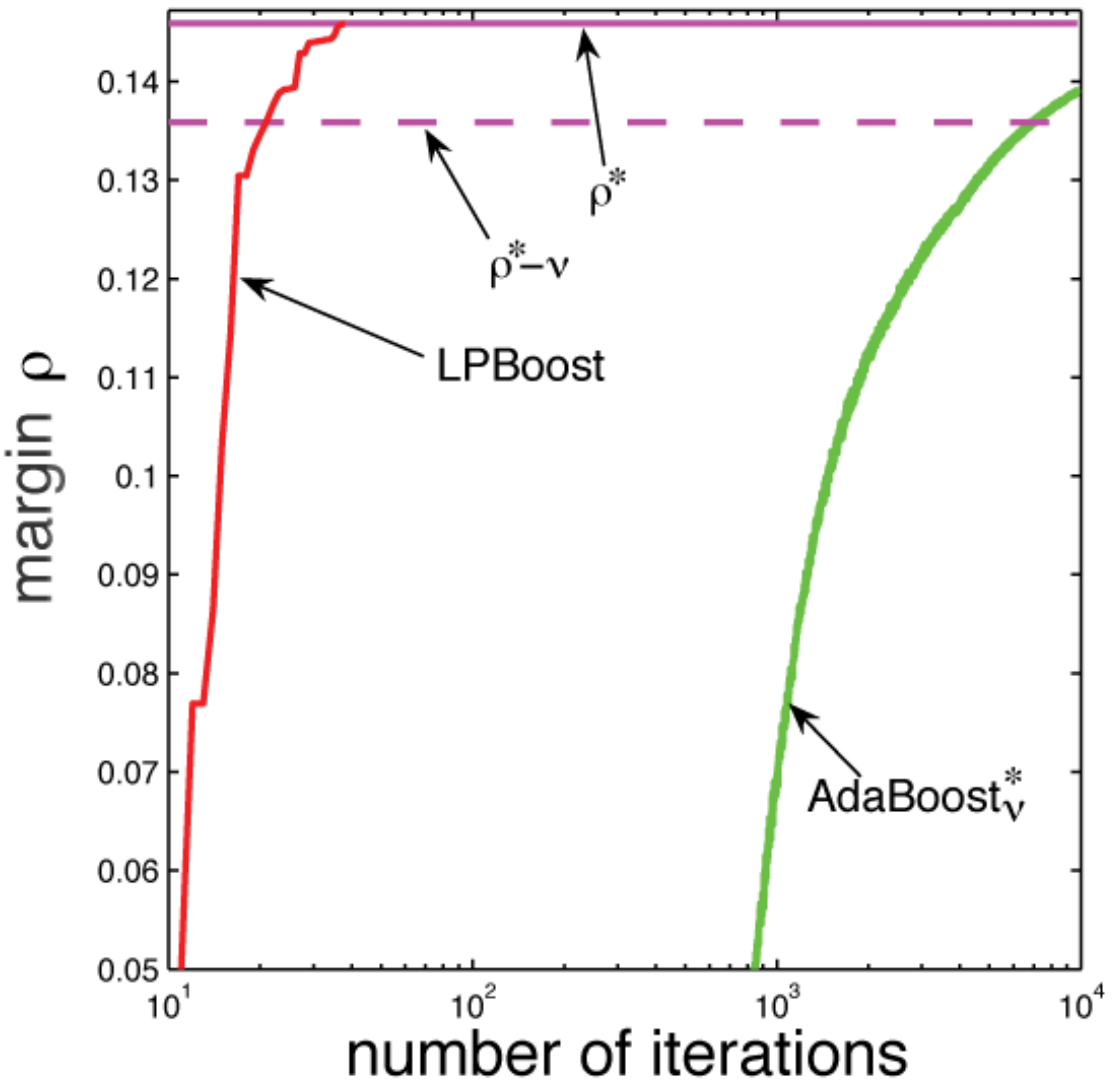


Cox-1 ($\nu = 0.01$)

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

Results

Illustrative Experiments



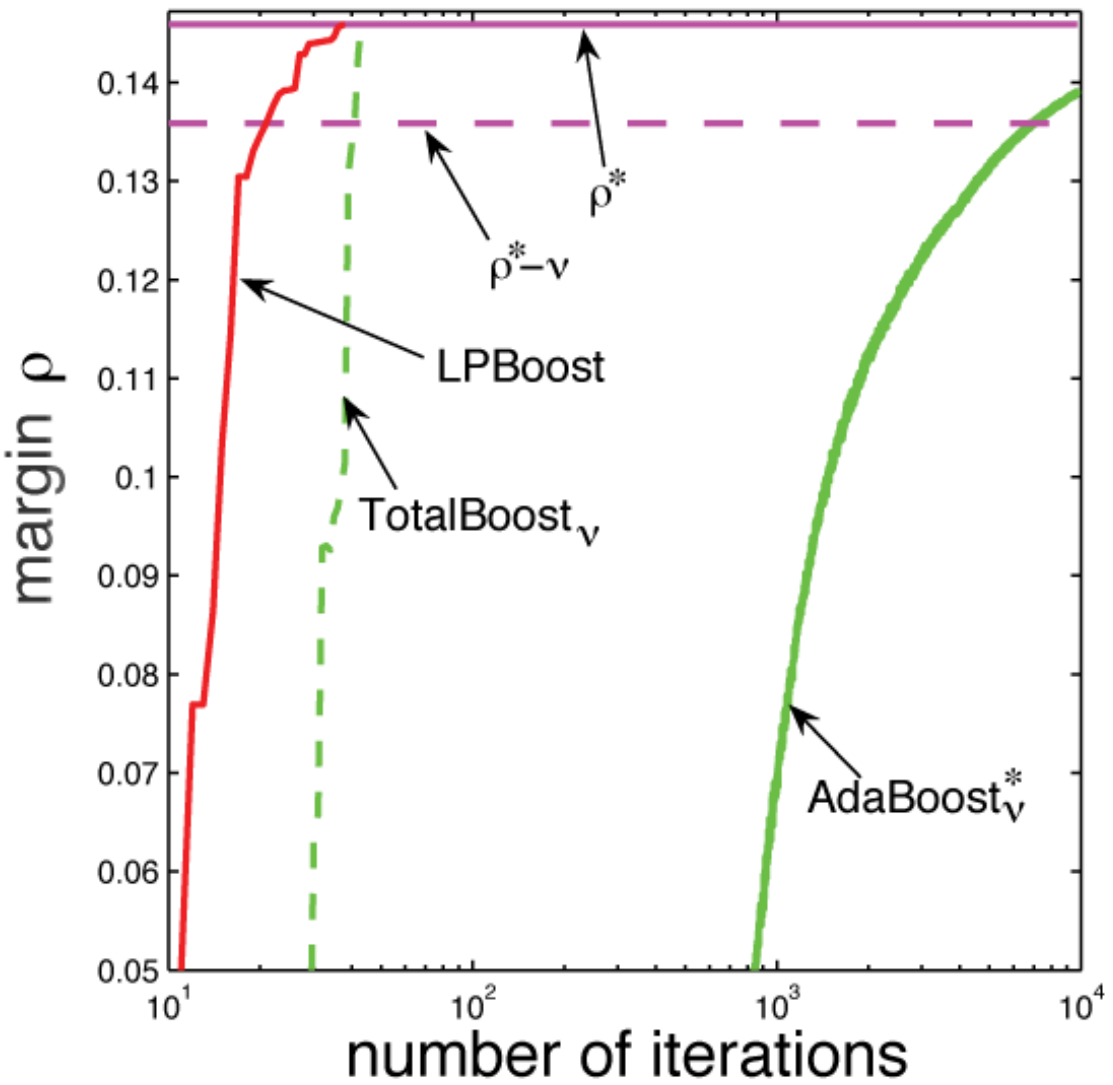
Cox-1 ($\nu = 0.01$)

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

Results

- Corrective algorithms very slow
- LPBoost & TotalBoost need few iterations
- AdaBoost* in between

Illustrative Experiments



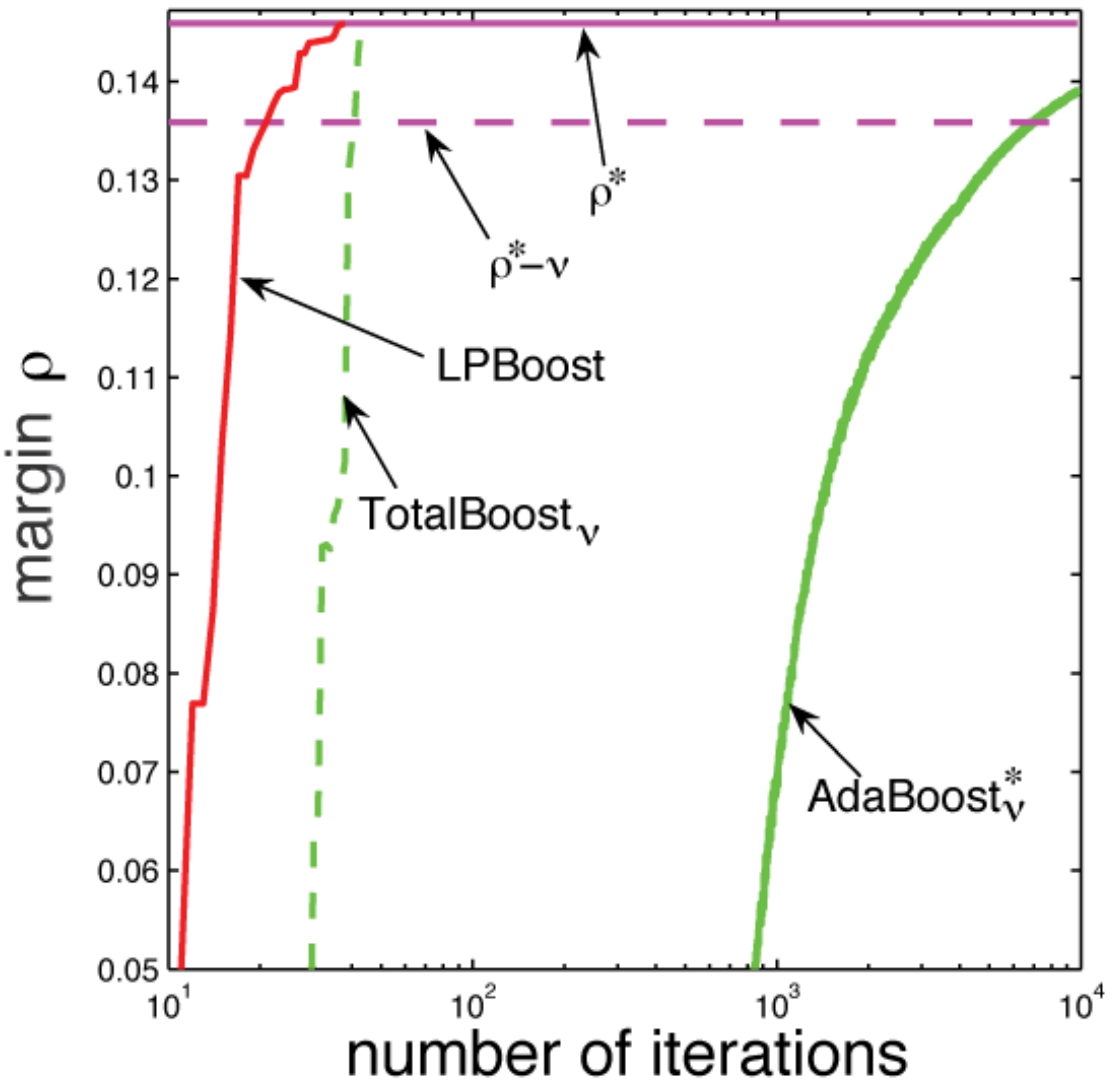
Cox-1 ($\nu = 0.01$)

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

Results

- Corrective algorithms very slow
- LPBoost & TotalBoost need few iterations
- AdaBoost$^*_\nu$ in between

Illustrative Experiments



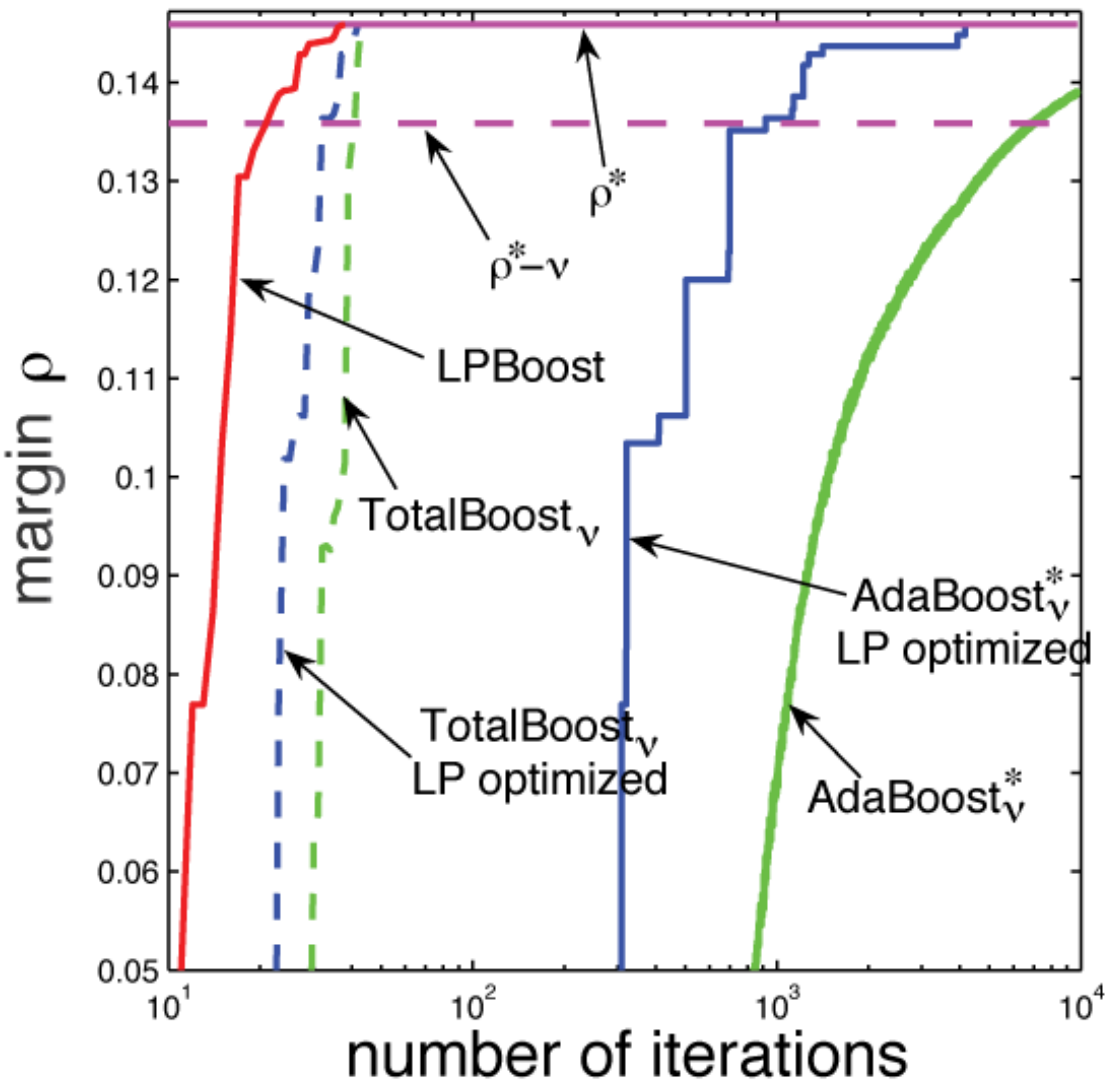
Cox-1 ($\nu = 0.01$)

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

Results

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

Illustrative Experiments



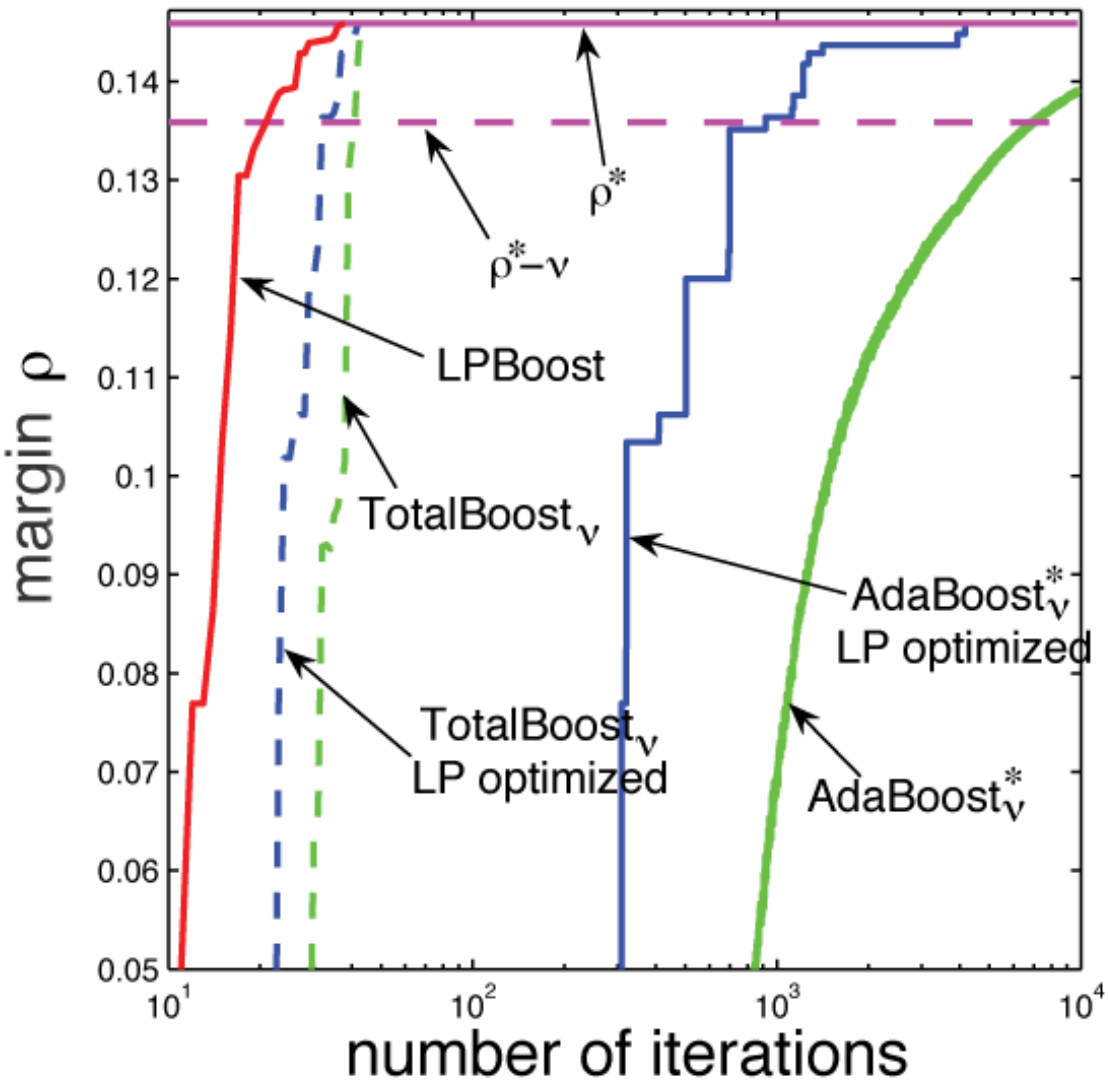
Cox-1 ($\nu = 0.01$)

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

Results

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

Illustrative Experiments



Cox-1 ($\nu = 0.01$)

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

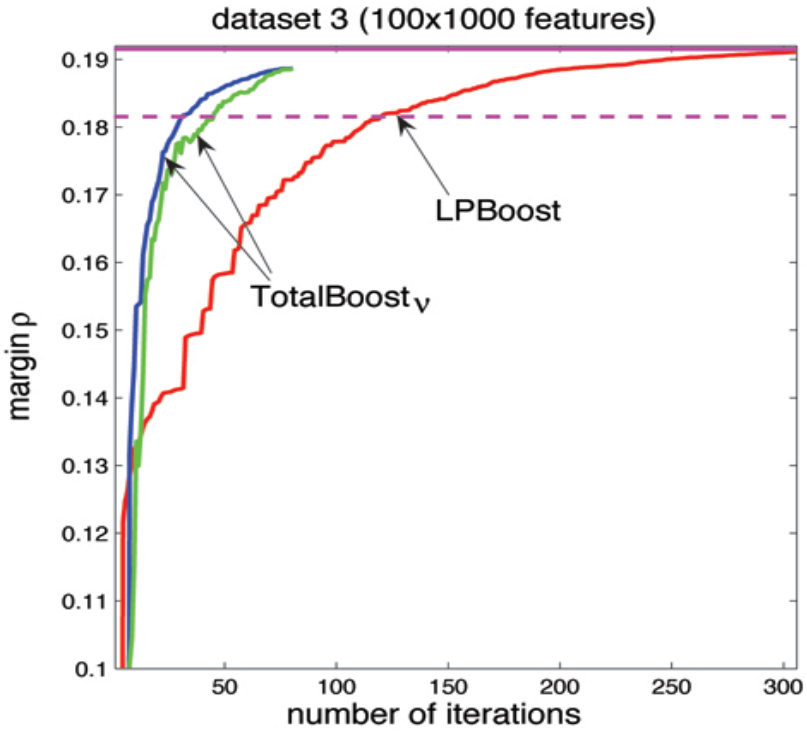
Results

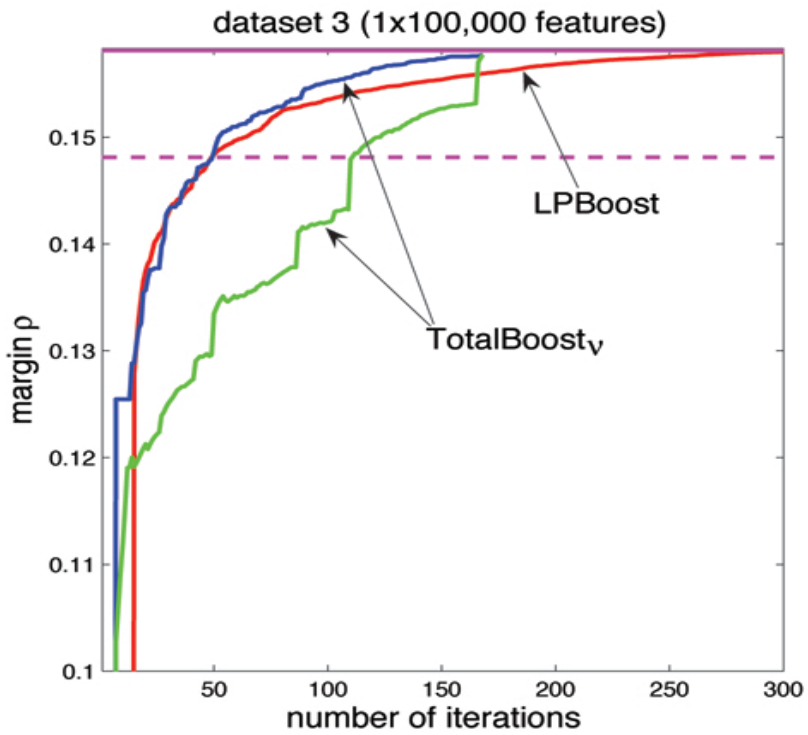
- Corrective algorithms very slow
- LPBoost & TotalBoost$_\nu$ need few iterations
- Initial speed crucially depends on ν

LPBoost May Perform Much Worse Than TotalBoost

- Identified cases where LPBoost converges considerably slower than TotalBoost$_\nu$
- Dataset is a series of artificial datasets of 1000 examples with varying number of features created as follows:
  - First generated N_1 random ± 1-valued features $x_1, \ldots, x_{N_1}$ and set the label of the examples as $y = \mathrm{sign}(x_1 + x_2 + x_3 + x_4 + x_5)$
  - Then duplicated each features N_2 times, perturbed the features by Gaussian noise with $\sigma = 0.1$, and clipped the feature values so that they lie in the interval [-1,1]
  - Considered different N_1, N_2, the total number of features is $N_2 \times N_1$

LPBoost performs worse for high dimensional data with many redundant features





LPBoost vs. TotalBoost$_\nu$ on two 100,000 dimensional datasets: [*left*] many redundant features ($N_1 = 1,000, N_2 = 100$) and [*right*] independent features ($N_1 = 100,000, N_2 = 1$). Show margin vs. number of iterations

Bound Not True for LPBoost

A counter example:

	Hypothesis No.					
	1	2	3	4	5	6
Examples	+	-	-	-	-	-
	+	-	-	-	-	-
	+	-	-	-	-	-
	+	-	-	-	-	-
	+	-	-	-	-	-
	-	+	-	-	-	+
	-	-	+	-	-	+
	-	-	-	+	-	+
	-	-	-	-	+	+

+: correct prediction -: incorrect prediction

TotalBoost averages hypothesis 1 and 6 (i.e. 2 iterations) to achieve maximum margin 0

Bound Not True for LPBoost

Hypothesis No.						Distribuition 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

LPBoost uses 5 iterations ((N+1)/2 iterations)
to achieve margin 0

Summarizing Boosting



- AdaBoost can be viewed as entropy projection
- TotalBoost projects based on all previous hypotheses
- Provably maximizes the margin
  - Theory: as fast as AdaBoost$^*_\nu$
  - Practice: much faster ($\approx$ LPBoost)
- Versatile techniques for proving iteration bounds
- Experiments corroborate our theory
  - Good for feature selection (not shown)
  - LPBoost may have problems of maximizing the margin
- Future: extension to the soft margin case

Part II

Semi-Infinite Linear Programming

Semi-Infinite Max-Margin LP



Primal

$$\begin{aligned} \max_{\boldsymbol{\alpha},\rho} \quad & \rho \\ \text{s.t.} \quad & U_n\boldsymbol{\alpha} \geq \rho \quad \forall n = 1,\ldots,N \\ & \boldsymbol{\alpha} \in \mathcal{P}^{|\mathcal{H}|} \end{aligned}$$

Dual

$$\begin{aligned} \min_{\gamma,\mathbf{d}} \quad & \gamma \\ \text{s.t.} \quad & U_h^\top \mathbf{d} \leq \gamma \quad \forall h \in \mathcal{H} \\ & \mathbf{d} \in \mathcal{P}^N \end{aligned}$$

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
- Primal solution very sparse
- Convergence theorems still hold

[G.R., Demiriz & Bennett, 2001; G.R., 2001]

Semi-Infinite Max-Margin LP



Primal

$$\begin{aligned} \max_{\boldsymbol{\alpha},\rho} \quad & \rho \\ \text{s.t.} \quad & U_n \boldsymbol{\alpha} \geq \rho \quad \forall n = 1, \ldots, N \\ & \boldsymbol{\alpha} \in \mathcal{P}^{|\mathcal{H}|} \end{aligned}$$

Dual

$$\begin{aligned} \min_{\gamma,\mathbf{d}} \quad & \gamma \\ \text{s.t.} \quad & U_h^\top \mathbf{d} \leq \gamma \quad \forall h \in \mathcal{H} \\ & \mathbf{d} \in \mathcal{P}^N \end{aligned}$$

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
- Primal solution very sparse
- Convergence theorems still hold

[G.R., Demiriz & Bennett, 2001; G.R., 2001]

Semi-Infinite Linear Programming



Special case of Semi-Infinite Linear Program

$$\begin{aligned} \min_{\mathbf{d}} \quad & \mathbf{c}^\top \mathbf{d} \\ \text{s.t.} \quad & A(h)^\top \mathbf{d} - b(h) \leq 0 \qquad \forall h \in \mathcal{H} \end{aligned}$$

Related to a Version of Chebyshev Approximation Problem (CAP)

$$\begin{aligned} \min_{\mathbf{d},\gamma} \quad & \gamma \\ \text{s.t.} \quad & A(h)^\top \mathbf{d} - b(h) \leq \gamma \qquad \forall h \in \mathcal{H} \end{aligned}$$

Reduction: (assume $\mathbf{d} \in \mathcal{P}^N$)

$$(A(h) - b(h)\mathbf{1} + \mathbf{c})^\top \mathbf{d} \leq \gamma \;\wedge\; \mathbf{c}^\top \mathbf{d} = \gamma \quad \Leftrightarrow \quad A(h)^\top \mathbf{d} - b(h) \leq 0$$

[Hettich & Kortanek, 1993; Reemtsen & Rückmann, 1998]

Semi-Infinite Linear Programming



Special case of Semi-Infinite Linear Program

$$\begin{aligned} \min_{\mathbf{d}} \quad & \mathbf{c}^\top \mathbf{d} \\ \text{s.t.} \quad & A(h)^\top \mathbf{d} - b(h) \leq 0 \qquad \forall h \in \mathcal{H} \end{aligned}$$

Related to a Version of Chebyshev Approximation Problem (CAP)

$$\begin{aligned} \min_{\mathbf{d},\gamma} \quad & \gamma \\ \text{s.t.} \quad & A(h)^\top \mathbf{d} - b(h) \leq \gamma \qquad \forall h \in \mathcal{H} \end{aligned}$$

Reduction: (assume $\mathbf{d} \in \mathcal{P}^N$)

$$(A(h) - b(h)\mathbf{1} + \mathbf{c})^\top \mathbf{d} \leq \gamma \;\wedge\; \mathbf{c}^\top \mathbf{d} = \gamma \quad \Leftrightarrow \quad A(h)^\top \mathbf{d} - b(h) \leq 0$$

[Hettich & Kortanek, 1993; Reemtsen & Rückmann, 1998]

Semi-Infinite Linear Programming



Special case of Semi-Infinite Linear Program

$$\begin{aligned} \min_{\mathbf{d}} \quad & \mathbf{c}^\top \mathbf{d} \\ \text{s.t.} \quad & A(h)^\top \mathbf{d} - b(h) \leq 0 \qquad \forall h \in \mathcal{H} \end{aligned}$$

Related to a Version of Chebyshev Approximation Problem (CAP)

$$\begin{aligned} \min_{\mathbf{d},\gamma} \quad & \gamma \\ \text{s.t.} \quad & A(h)^\top \mathbf{d} - b(h) \leq \gamma \qquad \forall h \in \mathcal{H} \end{aligned}$$

Reduction: (assume $\mathbf{d} \in \mathcal{P}^N$)

$$(A(h) - b(h)\mathbf{1} + \mathbf{c})^\top \mathbf{d} \leq \gamma \ \wedge \ \mathbf{c}^\top \mathbf{d} = \gamma \quad \Leftrightarrow \quad A(h)^\top \mathbf{d} - b(h) \leq 0$$

[Hettich & Kortanek, 1993; Reemtsen & Rückmann, 1998]

State of the art?

Many Algorithms ...

- Column Generation & Cutting Plane Algorithms ("LPBoost")
  - Exchange Methods to reduce complexity
  - Heuristics, no guaranteed convergence
- Extensions of Simplex-Algorithm
  - Bad worst case complexity
- Interior point methods
  - No convergence results for SILP case

All methods need efficient method to identify violated constraints.

No general convergence results

$\Rightarrow$ **Use Boosting-like methods!**

[Hettich & Kortanek, 1993; Reemtsen & Rückmann, 1998]

Why are SILPs important?



Many Applications ...

- Wavelet filter design
- Optimal control problems
- ℓ_1-regularized learning problems
  - Boosting
  - Multiple Kernel Learning
  - Structured Output Learning
- ...

[Hettich & Kortanek, 1993; Reemtsen & Rückmann, 1998]

Part III

Applications

Current Projects



Machine Learning

- Improved learning algorithms
  - Large scale (>10 millions of examples)
  - Structured outputs (segmentations, graphs, etc)
- Methods for understanding learning results
  - Helps to understand how biological mechanisms work
- Improved alignment algorithms
  - Accurately find micro-exons, alternative splicing events etc.

Genome Biology

- *Ab initio* gene finding
  - Prediction of (alternative) splicing
  - Recognition of transcription starts and stops
- Prediction of polymorphisms from genome tiling arrays
  - Prediction of SNPs, deletions & highly polymorphic regions
  - Predicting changes in gene products
- Other projects: secondary structure prediction...

Current Projects



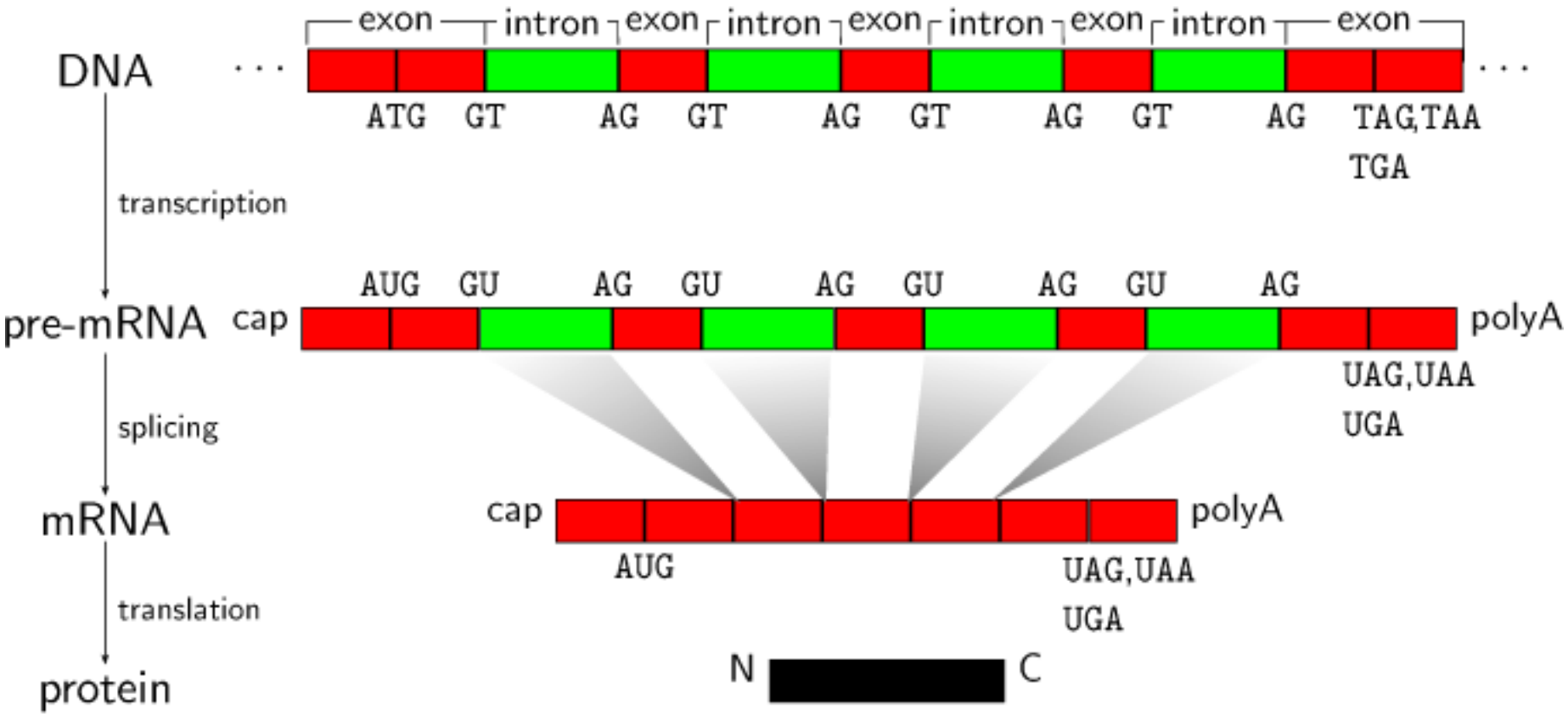
Genome Biology Machine Learning

- Improved learning algorithms
  - Large scale (>10 millions of examples)
  - Structured outputs (segmentations, graphs, etc)
- Methods for understanding learning results
  - Helps to understand how biological mechanisms work
- Improved alignment algorithms
  - Accurately find micro-exons, alternative splicing events etc.
- *Ab initio* gene finding
  - Prediction of (alternative) splicing
  - Recognition of transcription starts and stops
- Prediction of polymorphisms from genome tiling arrays
  - Prediction of SNPs, deletions & highly polymorphic regions
  - Predicting changes in gene products
- Other projects: secondary structure prediction...

Transcription, Splicing, Translation

Splice Sites: on the boundary between

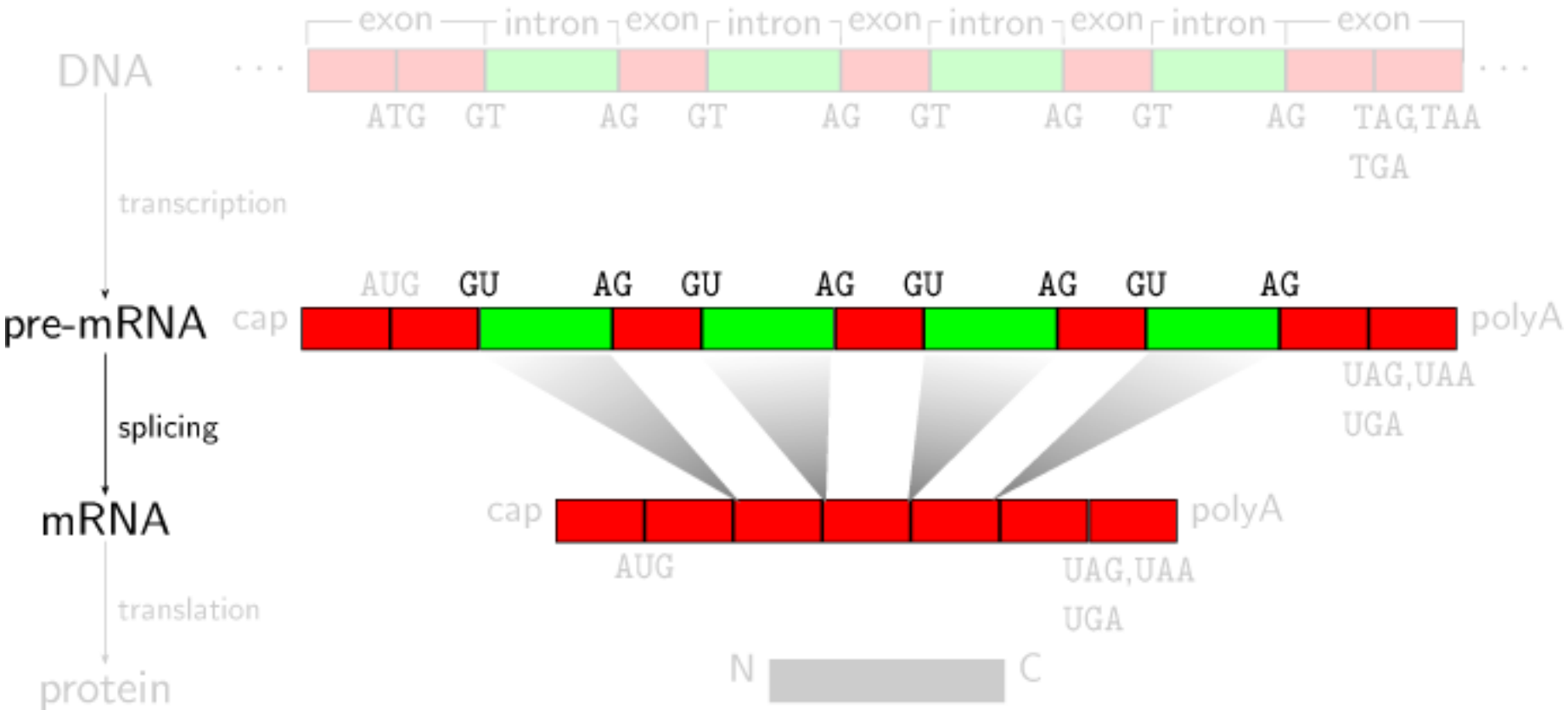
- **Exons (may code for protein) and**
- **Introns (non-coding)**



Transcription, Splicing, Translation

Splice Sites: on the boundary between

- **Exons (may code for protein) and**
- **Introns (non-coding)**



Ab initio Gene Finding



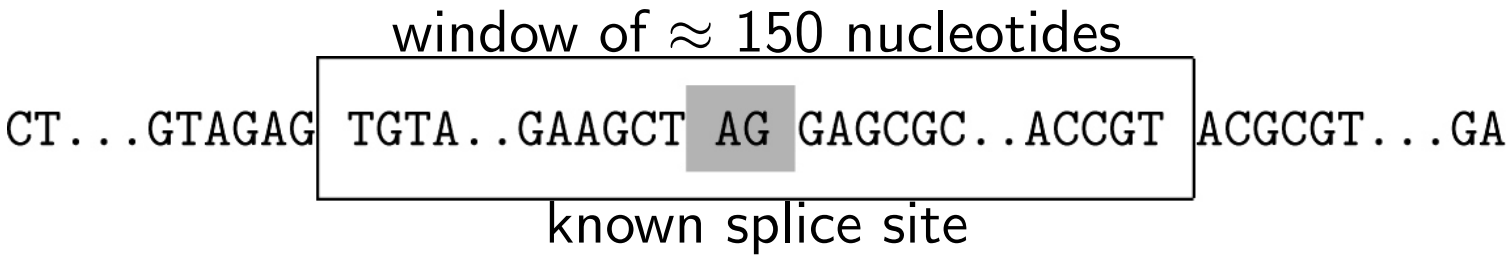
- Given the DNA one likes to predict the resulting mRNA and protein
- Requires very accurate identification of
  - Splice sites
  - Translation starts and stops
  - Promoters & polyA sites
  - Sites of regulation (transcription, splicing, etc.)
  - ...
- One needs a method to integrate single site predictions
  - Usually HMMs
  - Here: Learning Structured Outputs

Ab initio Gene Finding

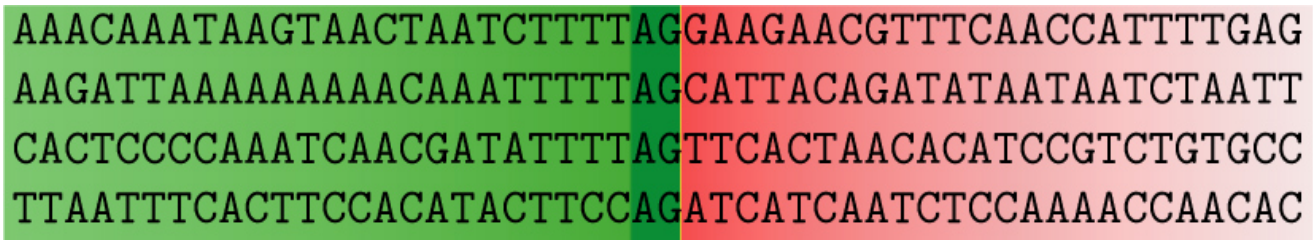


- Given the DNA one likes to predict the resulting mRNA and protein
- Requires very accurate identification of
  - Splice sites
  - Translation starts and stops
  - Promoters & polyA sites
  - Sites of regulation (transcription, splicing, etc.)
  - ...
- One needs a method to integrate single site predictions
  - Usually HMMs
  - Here: Learning Structured Outputs

2-class Splice Site Detection



- true sites: fixed window around a true splice site
- decoys sites: generated by shifting the window



⇒ Millions of examples from EST databases

⇒ Very accurate SVM-based predictors (ROC=99.7%)

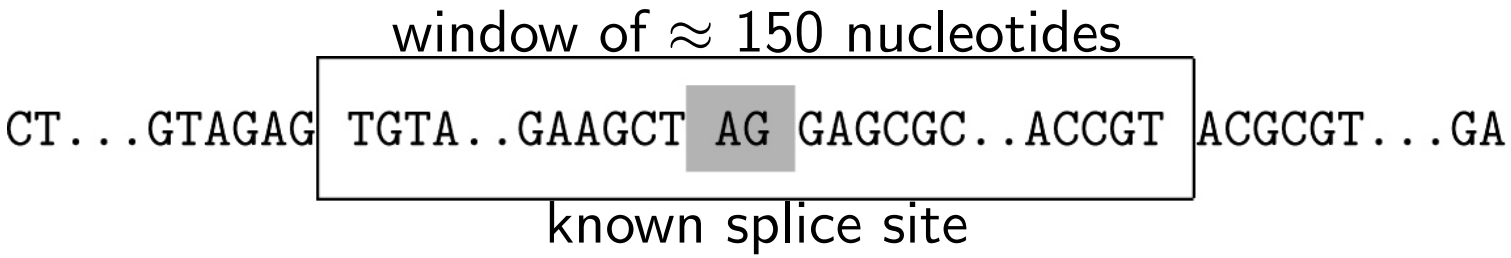
Biologists want to understand what is used for discrimination

⇒ Identify interesting positions

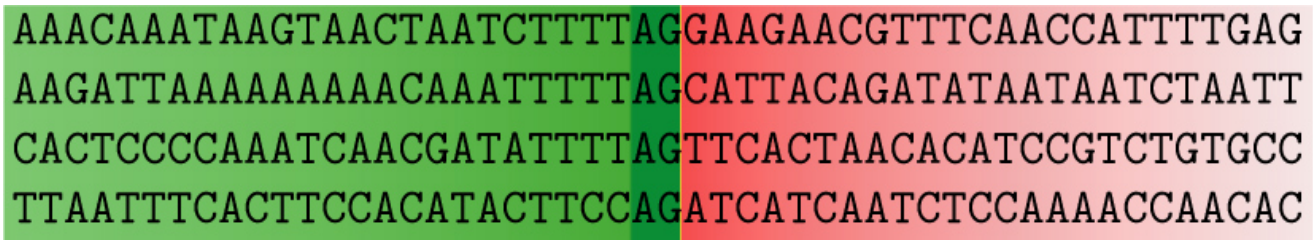
How?

[G.R. & Sonnenburg, 2004]

2-class Splice Site Detection



- true sites: fixed window around a true splice site
- decoys sites: generated by shifting the window



⇒ Millions of examples from EST databases

⇒ Very accurate SVM-based predictors (ROC=99.7%)

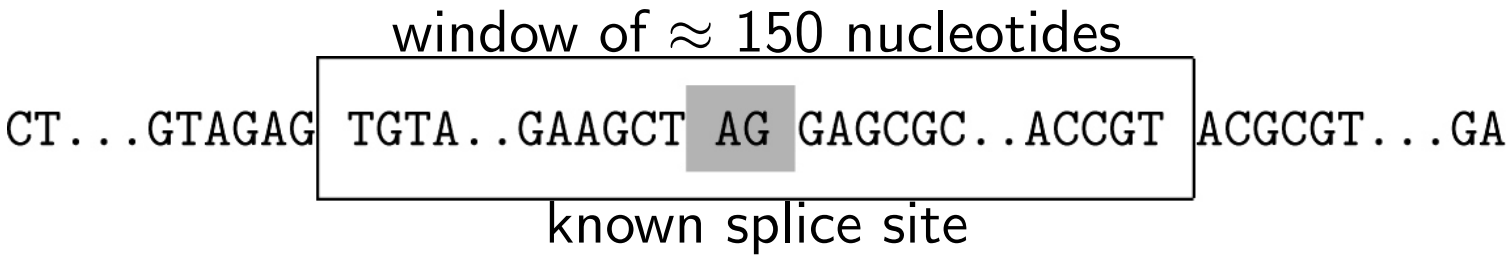
Biologists want to understand what is used for discrimination

⇒ Identify interesting positions

How?

[G.R. & Sonnenburg, 2004]

2-class Splice Site Detection



- true sites: fixed window around a true splice site
- decoys sites: generated by shifting the window

```
AAACAAATAAGTAACTAATCTTTTAGGAAGAACGTTTCAACCATTTTGAG
AAGATTAAAAAAAAACAAATTTTTAGCATTACAGATATAATAATCTAATT
CACTCCCCAAATCAACGATATTTTAGTTCACTAACACATCCGTCTGTGCC
TTAATTTCACTTCCACATACTTCCAGATCATCAATCTCCAAAACCAACAC
```

⇒ Millions of examples from EST databases

⇒ Very accurate SVM-based predictors (ROC=99.7%)

Biologists want to understand what is used for discrimination

⇒ Identify interesting positions

How?

[G.R. & Sonnenburg, 2004]

Multiple Kernel Learning: Motivation



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

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

- Now: Convex combination of kernels

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

- **useful:** Polynomial kernels of different degree, kernels on different domains
- **but:** How to learn weights β_j ?
  - $\Rightarrow$ Make it part of the SVM optimization!

MKL Optimization: Primal

$$
\begin{aligned}
\text{min} \quad & \frac{1}{2}\left(\sum_{j=1}^{M}\beta_j\|\mathbf{w}_j\|_2\right)^2 + C\sum_{i=1}^{N}\xi_n \\
\text{w.r.t.} \quad & \mathbf{w}=(\mathbf{w}_1,\ldots,\mathbf{w}_M), \mathbf{w}_j\in\mathbf{R}^{k_j}, \boldsymbol{\xi}\in\mathbf{R}_+^N, \boldsymbol{\beta}\in\mathbf{R}_+^M, b\in\mathbf{R} \\
\text{s.t.} \quad & y_i\left(\sum_{j=1}^{M}\beta_j{\mathbf{w}_j}^\top\mathbf{x}_{i,j}+b\right)\geq 1-\xi_i, \forall i=1,\ldots,N \\
& \sum_{j=1}^{M}\beta_j = 1
\end{aligned}
$$

Properties: equivalent to linear SVM for $M = 1$; solution sparse in "blocks"; each block j corresponds to one kernel

[Lanckriet et al., 2004; Bach et al., 2004]

MKL Optimization: Dual

$$
\begin{aligned}
\text{min} \quad & \frac{1}{2}\gamma^2 - \sum_{i=1}^{N} \alpha_i \\
\text{w.r.t.} \quad & \gamma \in \mathbf{R}, \boldsymbol{\alpha} \in \mathbf{R}^N \\
\text{s.t.} \quad & 0 \leq \boldsymbol{\alpha} \leq C, \sum_{i=1}^{N} \alpha_i y_i = 0 \\
& \underbrace{\sum_{r=1}^{N}\sum_{s=1}^{N} \alpha_r \alpha_s y_r y_s K_j(\mathbf{x}_r, \mathbf{x}_s)}_{=:S_j(\boldsymbol{\alpha})} - \gamma^2 \leq 0, \ \forall j = 1, \ldots, M
\end{aligned}
$$

[Lanckriet et al., 2004; Bach et al., 2004]

MKL Optimization: Reformulation as SILP



$$\begin{aligned}
\max \quad & \theta \\
\text{w.r.t.} \quad & \theta \in \mathbf{R}, \boldsymbol{\beta} \in \mathbf{R}_+^M \text{ with } \sum_{j=1}^{M} \beta_j = 1 \\
\text{s.t.} \quad & \sum_{j=1}^{M} \beta_j \left(\frac{1}{2} S_j(\boldsymbol{\alpha}) - \sum_{i=1}^{N} \alpha_i \right) \geq \theta \\
& \text{for all } \boldsymbol{\alpha} \text{ with } 0 \leq \boldsymbol{\alpha} \leq C \text{ and } \sum_{i=1}^{N} y_i \alpha_i = 0
\end{aligned}$$

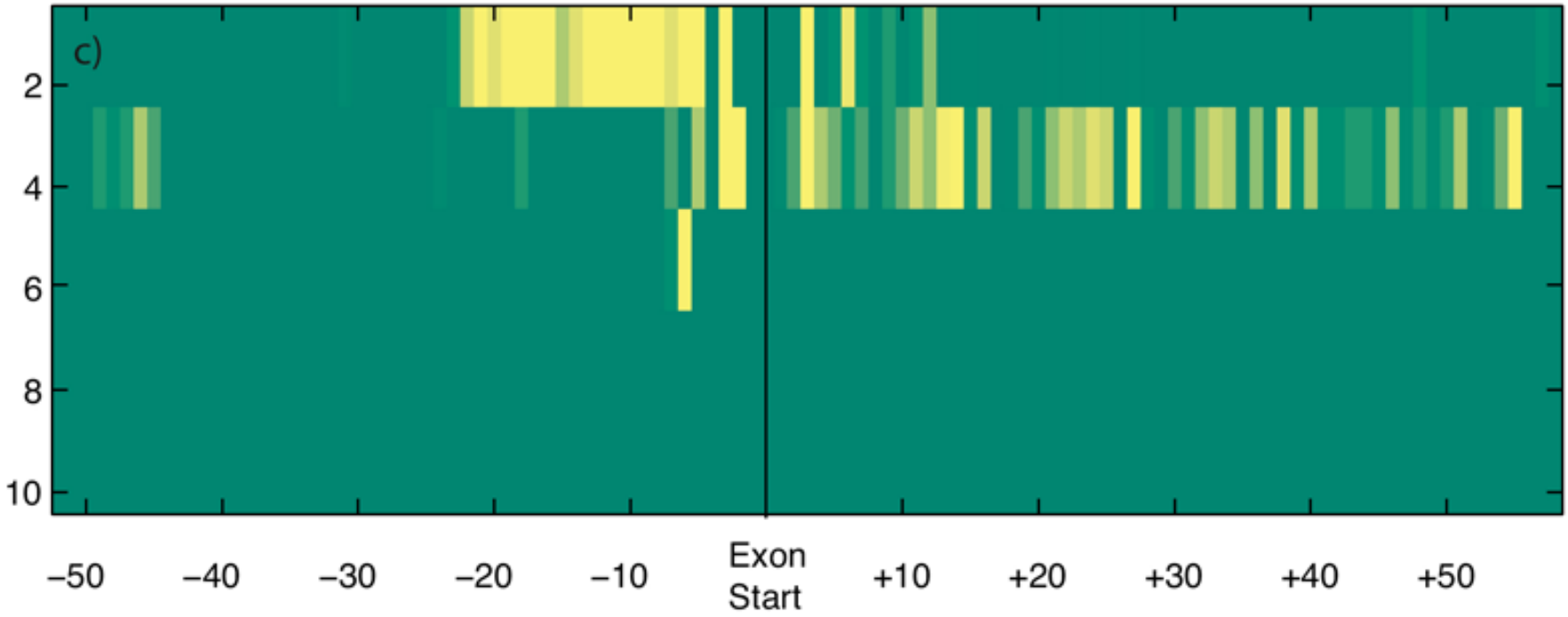
Use Boosting techniques to solve SILP

- Finding corresponds to finding active constraint:

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

$\Rightarrow$ Solve SVM for some given $\boldsymbol{\beta}$'s [Sonnenburg, G.R., Schäfer, 2005]

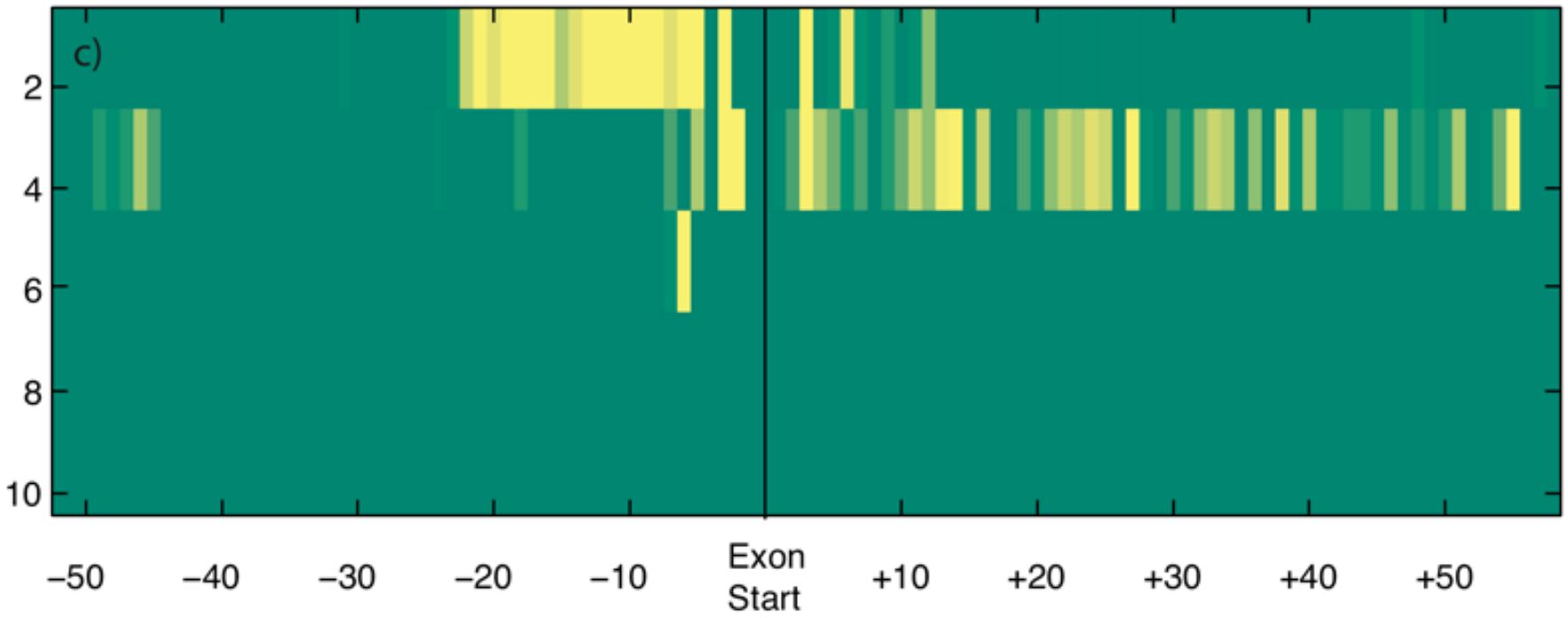
Application to Splice Detection



Back to gene finding . . .

[G.R., Sonnenburg, Schäfer, Schölkopf, 2006]

Application to Splice Detection



Back to gene finding . . .

[G.R., Sonnenburg, Schäfer, Schölkopf, 2006]

Learning Structured Outputs



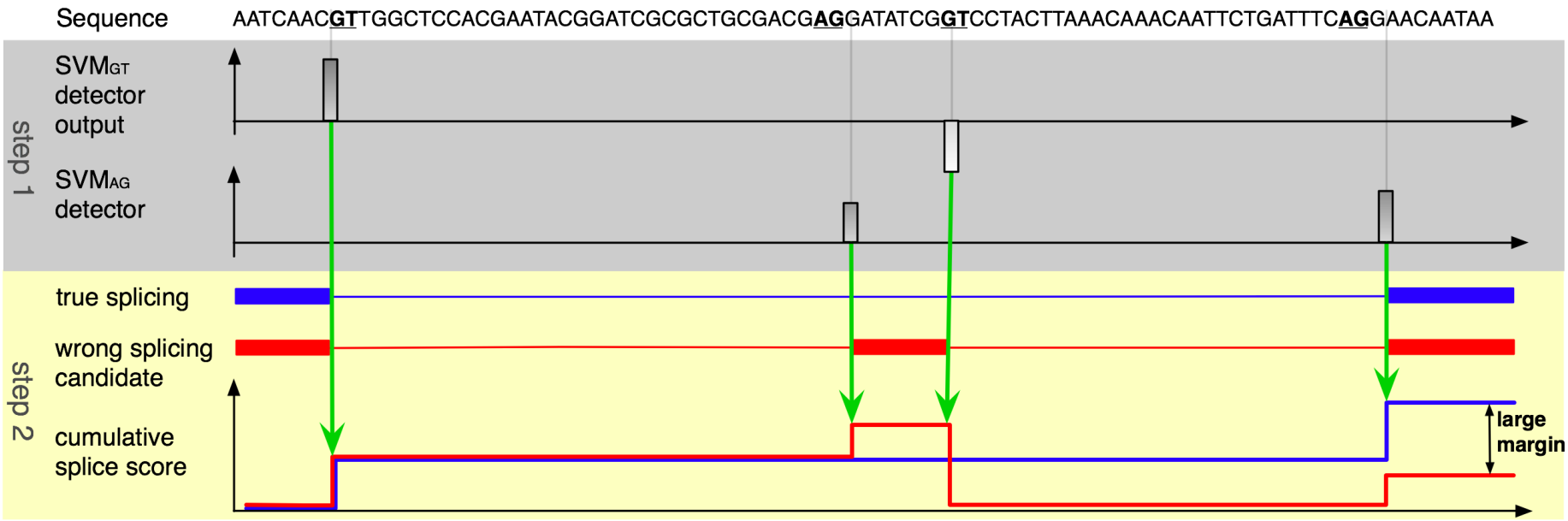
- Learn function S_θ to score "structures" $\mathbf{s}$ (linear in $\theta \in \mathbb{R}^d$)
- Idea: Score true structure much higher than all other structures
- Solve optimization problem:

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

  $\Rightarrow$ Exponentially many constraints

  $\Rightarrow$ Find violating constraints with dynamic programming
- Chose appropriate $\mathbf{P}$
  - Linear $\Rightarrow$ SILP techniques
  - Quadratic $\Rightarrow \mathcal{O}(R^2/\epsilon^2)$ iterations [Tsochantaridis et al., 2004]

Large Margin Splice Form Prediction



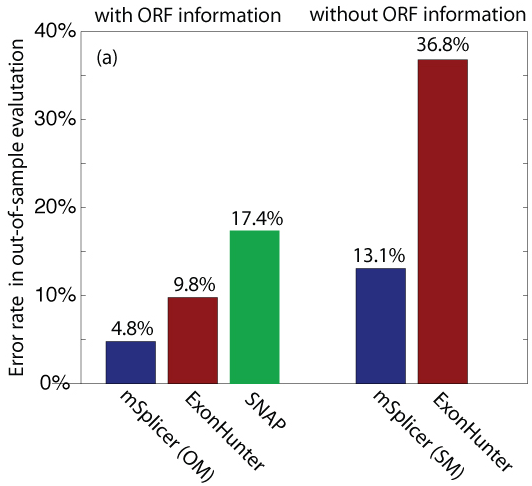
- Simplified Model: Score for splicing $\mathbf{s} = \{(p_j, q_j)\}_{j=1}^{J}$):

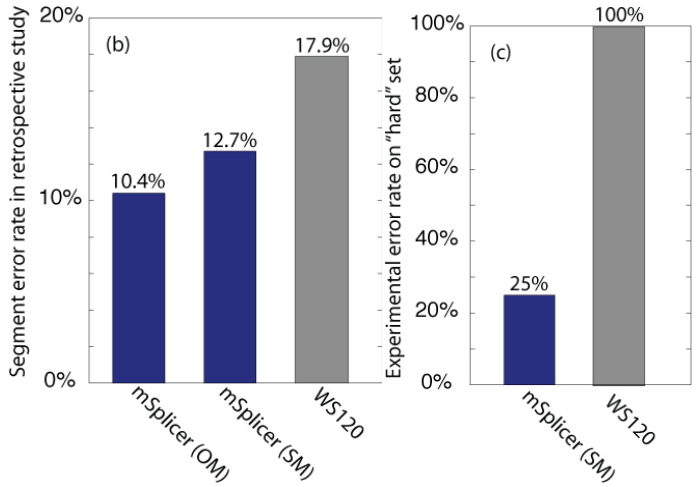
$$S(\mathbf{s}) := \underbrace{\sum_{j=1}^{J-1} S_{GT}(f_j^{GT}) + \sum_{j=2}^{J} S_{AG}(f_j^{AG})}_{\text{Splice signals}} + \underbrace{\sum_{j=1}^{J-1} S_{L_I}(p_{j+1} - q_j) + \sum_{j=1}^{J} S_{L_E}(q_j - p_j)}_{\text{Segment lengths}}$$

- Finding the best splicing: use dynamic programming
- Free parameters: functions S_{GT}, S_{AG}, S_{L_E} and S_{L_I}

Results for *Caenorhabditis elegans*

- Out-of-sample accuracy (≈1100 cDNAs):
  - Two versions: coding and mixed mRNA
  - 87-95% of all splice forms correct
  - Much lower error rates than other methods
- Comparison with unconfirmed gene models:
  - Only 37-50% agreement with annotation!
  - Retrospective comparison . . .
    . . . Using newer sequences
    . . . Shows improved accuracy
- Exp. validation in 20 disagreeing cases:
  - 75% of our predictions were correct
  - Annotation was never correct

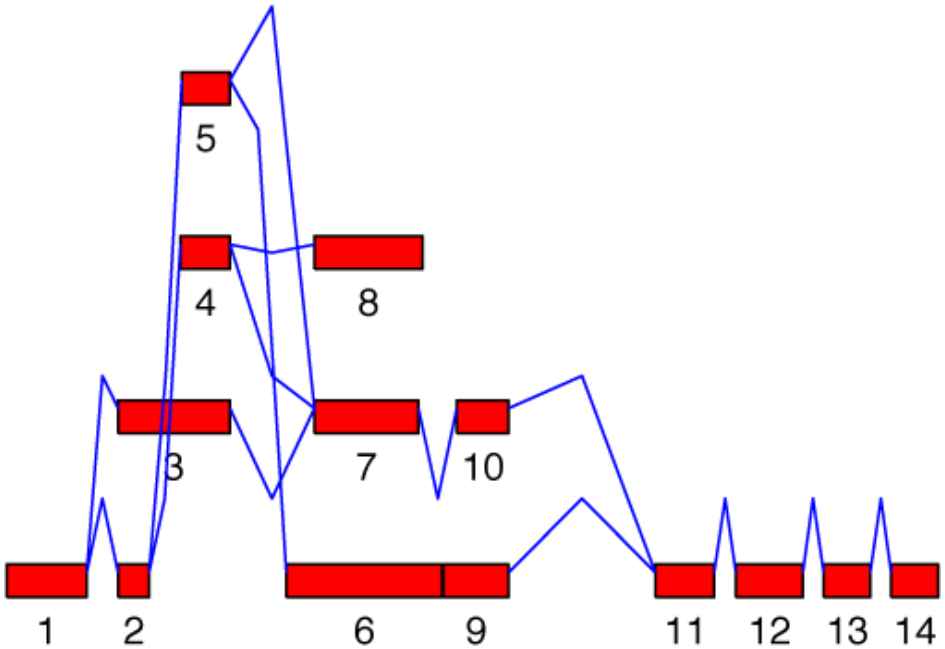




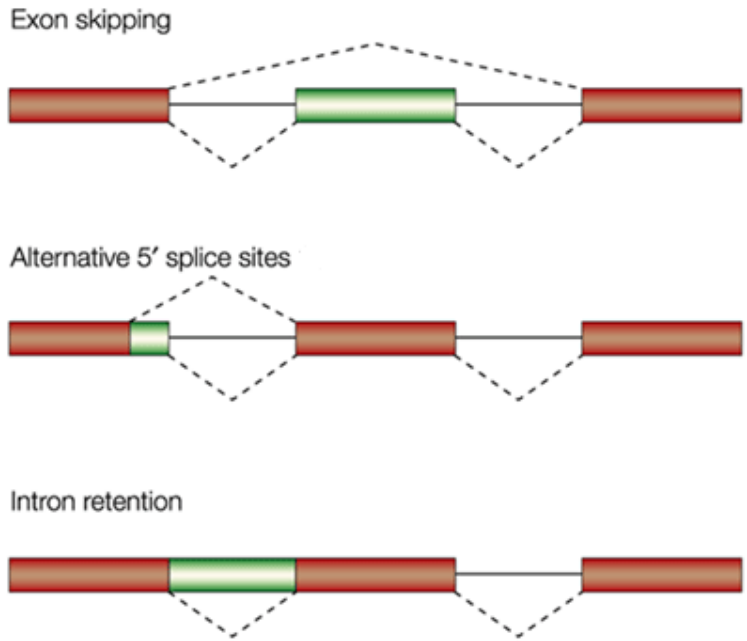
[G.R. et al., G.R. & Sonnenburg, 2006]

Challenges for Machine Learning

Reality: Alternative Splicing



Simple classes:



- Predicting the simple cases is not enough

$\Rightarrow$ need to predict the gene structure

- Difficult learning setting:
  - Input: DNA sequence
  - Output: Splice-graph (vertexes & edges unknown)

Summary



- Semi-infinite or huge linear programs frequently appear in machine learning
- Iterative methods require efficient algorithm to identify violated constraints
  - Boosting: Weak learner
  - Multiple kernel learning: Single kernel SVM
  - Structure Learning: Dynamic programming
- TotalBoost solves SILPs and combines benefits:
  - Convergence guarantee: $\mathcal{O}(\ln N/\nu^2)$
  - About as fast as LPBoost, much faster than AdaBoost
  - Does not degrade in case of redundant features
  - Proof technique applies to other divergences
- Open problems
  - Remove dependence on ν?
  - Is there a lower bound or a faster algorithm?

The End

- Slides with references will be available at `http://www.fml.mpg.de/~raetsch/alt06`
- Collaborators for this work:
  Kristin Bennett, Ayhan Demiriz, Jun Liao, Sebastian Mika, Klaus-Robert Müller, Takashi Onoda, Bernhard Schölkopf, Sören Sonnenburg, Ralf Sommer, Jagan Srinivasan, Manfred Warmuth
- **Thank You!**


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


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.


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.


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.


G. Rätsch.
Robust Boosting via Convex Optimization.

PhD thesis, University of Potsdam, Neues Palais 10, 14469 Potsdam, Germany, October 2001.


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


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


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


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.