An Introduction to Boosting
Part III

Gunnar Rätsch

Friedrich Miescher Laboratory
of the Max Planck Society

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

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

Contents of this Course

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

Slides available from the wiki and from
http://www.fml.mpg.de/~raetsch/boostintro

7. Boosting vs. SVMs



Hill Inlet, Whitsunday Island

Main Sources:

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

SVM's Mathematical Program

The SVM minimization of

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

reformulate as maximization of the margin ρ

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

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

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

Boosting's Mathematical Program



- master hypothesis

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

- base hypotheses h_t produced by the base algorithm.

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

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

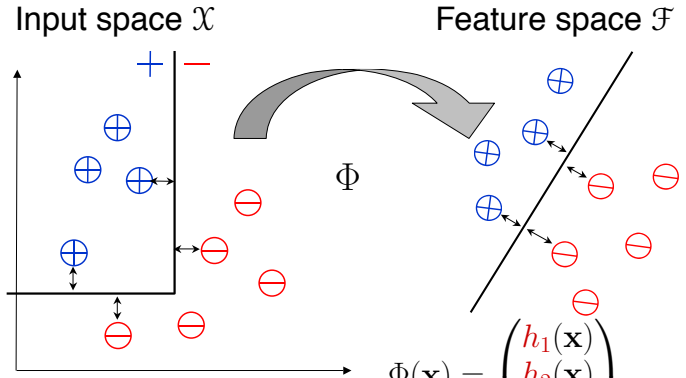
$$\text{subject to} \quad y_n \sum_{j=1}^{J} w_j h_j(\mathbf{x}_n) \geq \rho \quad \text{for} \quad n = 1, \ldots, N \qquad (3)$$

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

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

Large Margin and Linear Separation





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

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

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

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

Even more explicit



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

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

and a kernel

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

which could in principle be used for SVM learning.

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

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

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

About Norms



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

What happens in the long run?

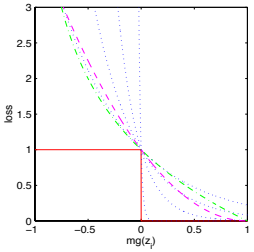
- Explicit expression for $d_n^{(t+1)}$:

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

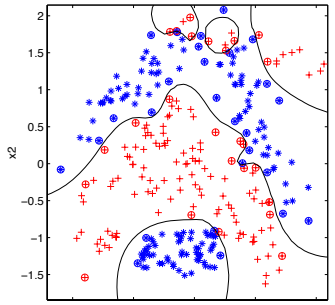
  $\rightsquigarrow$ **Soft-Max Function** with parameter $\|\boldsymbol{\alpha}^{(t)}\|_1$

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

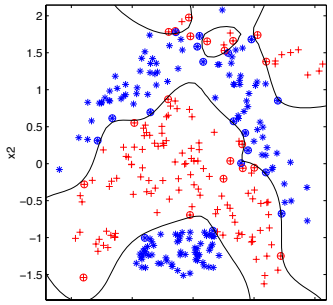
$\rightsquigarrow$ the d's concentrate on a **few** difficult patterns
$\rightarrow$ **"Support Patters"**



Support Patterns vs. Support Vectors



AdaBoost's decision line



SVM's decision line

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

Boosting vs. SVMs I

- SVMs

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

- Boosting

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

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

Representer Theorem for Kernel Algorithms

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

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

Then for any N examples $(\mathbf{x}_1, y_1), \ldots, (x_N, y_N)$ and any mapping Φ, then **any solution** $\mathbf{w}^*$ of (4) can be expressed by

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

for some $\alpha_n \in \mathbf{R}$, $n = 1, \ldots, N$.
Only ℓ_2-norm regularizers have this property!

Representer Theorem for Boosting Algs



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

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

Then for any N example $(\mathbf{x}_1, y_1), \ldots, (x_N, y_N)$ and any compact hypothesis set $\mathcal{H}$, there **exist solutions** $f_{\mathbf{w}^*}$ of (5) that can be expressed by

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

Most of the w's can be made zero!
The ℓ_1-norm is the only convex regularizer that has this property!

Boosting vs. SVMs II



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

Ingredients of Boosting



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

8. Improving Robustness & Extensions



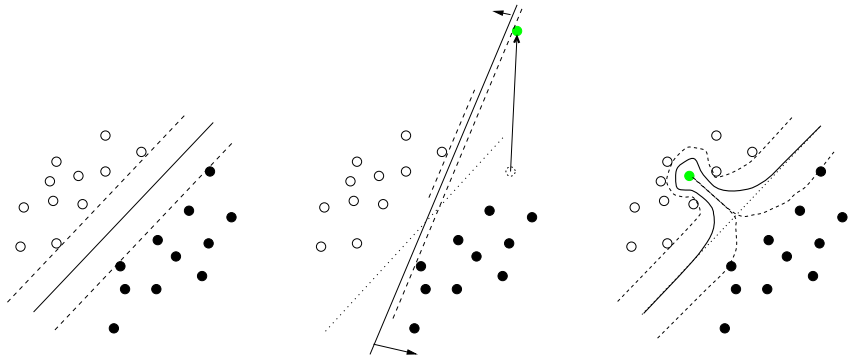


Hill Inlet, Whitsunday Island

Main Sources:

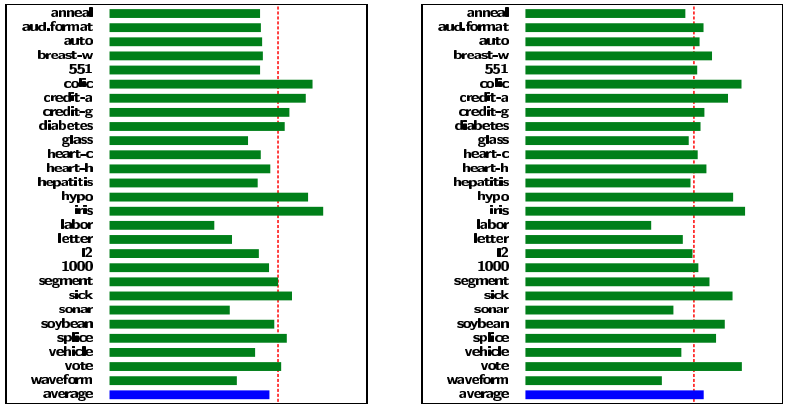
- Rätsch et al. 1998, 2000, 2001

Hard-Margin Classification



- The problem of finding a maximum margin "hyper-plane" on reliable data (left), data with outlier (middle) and a mislabeled pattern (right). The hard margin implies **noise sensitivity**.

Experiments on noisy data



Comparison on UCI repository data (Quinlan (1998)): Left 5% and right 10% label noise

How to get regularized versions of Boosting?

- Under certain conditions we want to ignore
  - Outliers and
  - mislabeled patterns

$\Rightarrow$ Change AdaBoost's Error Function

$\Rightarrow$ AdaBoost$_{Reg}$ (Rätsch et al., 1998)

$\Rightarrow$ DOOM (Mason et al., 1998)

$\Rightarrow$ Logit-Boost (Friedman et al., 1998)

$\Rightarrow$ Brown-Boost (Freund, 1999)

$\Rightarrow$ **Soft Margin** (like in SVM's)

- Extend the underlying LP to soft margins

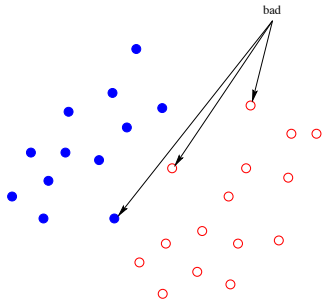
  $\Rightarrow$ ν-AdaBoost (Rätsch et al., 2000)

AdaBoost with Soft Margins

- Define a Soft Margin

$$\tilde{\rho}_n(\boldsymbol{\alpha}) = \rho_n(\boldsymbol{\alpha}) + \zeta_n,$$

  - where ζ_n is the amount of uncertainty in example $(\mathbf{x}_n, y_n)$
- Illustration



xx

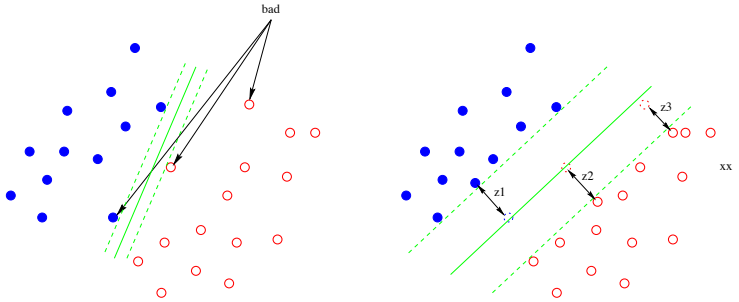
AdaBoost with Soft Margins



- Define a Soft Margin

$$\tilde{\rho}_n(\boldsymbol{\alpha}) = \rho_n(\boldsymbol{\alpha}) + \zeta_n$$

  - where ζ_n is the amount of uncertainty in pattern $\mathbf{z}_n$
- Illustration



AdaBoost with Soft Margins



- Once we have defined the uncertainty measure ζ_n, we can easily get a new regularized Boosting algorithm.

  $\Rightarrow$ Improve the Error Function by plugging-in the Soft Margin

$$\tilde{G}(\boldsymbol{\alpha}) = \sum_{n=1}^{N} \exp\left\{-\|\boldsymbol{\alpha}\|_1 \tilde{\rho}_n(\boldsymbol{\alpha})\right\}$$

$$d_n^{t+1} = \frac{\partial \tilde{G}(\boldsymbol{\alpha})}{\partial f_{\boldsymbol{\alpha}}(\mathbf{x}_n))}$$

$$\alpha_t = \operatorname*{argmin}_{\alpha_t \geq 0} \tilde{G}(\boldsymbol{\alpha})$$

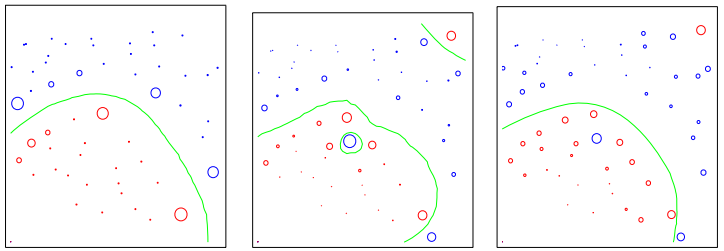
AdaBoost$_{Reg}$ – Reducing the Influence



- How can we know which patterns are unreliable?

  AdaBoost focuses on difficult-to-learn patterns by assigning high pattern weights d_n that we can exploit. Hence, we define the Influence of a pattern

$$\mu_n^t = \sum_{r=1}^{t} \frac{\alpha_r}{\|\boldsymbol{\alpha}\|_1} d_n^r \qquad \zeta_n = C(\mu_n^t)^2$$



AdaBoost without "Noise" **AdaBoost with "Noise"** **AdaBoost$_{Reg}$ with "Noise"**

AdaBoost$_{Reg}$



Positive:

- first algorithm that addresses overfitting in Boosting
- much improved results

Negative:

- Modification on algorithmic level
- hard to analyze
  - which optimization problem is solved
  - no generalization results

Idea:

- go back to beginning and redesign optimization problem
- use convergence results for leveraging
- apply margin bounds

ν-LP-Classification



Input: - N examples $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$
- hypothesis set $\mathcal{H}$ with J elements

- Introduce Slack-Variables ξ_n to the margin constraints
- Solve the Linear Program to find the hypotheses weights $\boldsymbol{\alpha}$:

$$\text{Maximize } \nu\rho - \frac{1}{N}\sum_{n=1}^{N}\xi_n \text{ under the constraints}$$

- $\rho_n(\boldsymbol{\alpha}) \geq \rho - \xi_n \qquad n = 1, \ldots, N$
- $\xi_n \geq 0 \qquad n = 1, \ldots, N$
- $\alpha_j \geq 0 \qquad j = 1, \ldots, J$
- $\|\boldsymbol{\alpha}\|_1 = 1$

Result: Empirical margin error is about ν (so-called ν-trick)

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



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

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

Observe:

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

How to get regularized versions of Boosting?



- Extend/define the underlying optimization problem
  - . . . e.g. by incorporating some regularization . . .
- (re-)construct a new algorithm
  - . . . e.g. using Barrier (Rätsch et al., 2000) or Column-Generation techniques (Bennett et al., 2000). . .
- prove the convergence using tools from optimization

underlying opt. problem	Construct the new Algorithm using		
	AdaBoost	Barrier Meth.	CG-AdaBoost
LP-Classification	AdaBoost*		CG-AdaBoost
ν-LP-Classification	RoBoost	C-Bar-Lev	ν-CG-AdaBoost
ν-LP-Regression	+	ε-Boost	ν-CG-Regression
ν-LP One-class	-	One-Lev	1-Class CG
Multi-class LP	AdaBoost.M2	. . .	+
. . .	. . .	. . .	. . .

Barrier Optimization



Constraint Optimization:

$$\min_{\boldsymbol{\theta}} \; g(\boldsymbol{\theta}) \quad \text{s.t. } c_n(\boldsymbol{\theta}) \geq 0, \; n = 1, \ldots, N \qquad \min_{\boldsymbol{\alpha},\rho} \; -\rho \quad \text{s.t. } \rho_n(\boldsymbol{\alpha}) - \rho \geq 0, \; n = 1, \ldots, N$$

$$E_\beta(\boldsymbol{\theta}) = g(\boldsymbol{\theta}) + \sum_{n=1}^{N}\kappa_\beta(c_n(\boldsymbol{\theta})) \qquad E_\beta(\boldsymbol{\alpha}, \rho) = -\rho + \sum_{n=1}^{N}\kappa_\beta(\rho_N(\boldsymbol{\alpha}) - \rho)$$

Convergence: $\lim_{\beta\to 0} \operatorname{argmin}_{\boldsymbol{\theta}} E_\beta(\boldsymbol{\theta}) = \boldsymbol{\theta}^*$ $\rightsquigarrow$ solution to problem

Barrierfunctions: $\kappa_\beta(t) = -\beta\log(t)$ $\rightsquigarrow$ Log-Barrier $\rightsquigarrow$ Interior Point
$\kappa_\beta(t) = \beta\exp(-t/\beta)$ $\rightsquigarrow$ Exp-Barrier $\rightsquigarrow$ Boosting

ν-LP vs C-LP



- Minimize $-\nu\rho + \frac{1}{N}\sum_{n=1}^{N}\xi_n$ under the constraints

$$\begin{aligned}
&\rho_n(\boldsymbol{\alpha}) \geq \rho - \xi_n \qquad n = 1,\ldots,N\\
&\xi_n \geq 0 \qquad n = 1,\ldots,N\\
&\alpha_j \geq 0 \qquad j = 1,\ldots,J\\
&\textstyle\sum_j \alpha_j = 1
\end{aligned}$$

- Minimize $\sum_j \alpha_j + \frac{1}{N}\sum_{n=1}^{N}\xi_n$ under the constraints

$$\begin{aligned}
&\rho_n(\boldsymbol{\alpha}) \geq 1 - \xi_n \qquad n = 1,\ldots,N\\
&\xi_n \geq 0 \qquad n = 1,\ldots,N\\
&\alpha_j \geq 0 \qquad j = 1,\ldots,J
\end{aligned}$$

C-LP with Barriers I



- Minimize $\sum_j \alpha_j + \frac{1}{N}\sum_{n=1}^{N}\xi_n$ under the constraints

$$\begin{aligned}
&\rho_n(\boldsymbol{\alpha}) \geq 1 - \xi_n \qquad n = 1,\ldots,N\\
&\xi_n \geq 0 \qquad n = 1,\ldots,N\\
&\alpha_j \geq 0 \qquad j = 1,\ldots,J
\end{aligned}$$

- Barrier objective s.t. $\boldsymbol{\alpha} \geq \mathbf{0}$

$$\begin{aligned}
E_\beta(\boldsymbol{\alpha},\boldsymbol{\xi}) \;=\; &\sum_j \alpha_j + \frac{1}{N}\sum_{n=1}^{N}\xi_n +\\
&+\beta\sum_n \exp\left(-\frac{\xi_n}{\beta}\right)\\
&+\beta\sum_n \exp\left(-\frac{\rho_n(\boldsymbol{\alpha}) - 1 + \xi_n}{\beta}\right)
\end{aligned}$$

C-LP with Barriers II



- Barrier objective s.t. $\boldsymbol{\alpha} \geq \mathbf{0}$

$$\begin{aligned}
E_\beta(\boldsymbol{\alpha},\boldsymbol{\xi}) \;=\; &\sum_j \alpha_j + \frac{1}{N}\sum_{n=1}^{N}\xi_n +\\
&+\beta\sum_n \exp\left(-\frac{\xi_n}{\beta}\right)\\
&+\beta\sum_n \exp\left(-\frac{\rho_n(\boldsymbol{\alpha}) - 1 + \xi_n}{\beta}\right)
\end{aligned}$$

- Determine optimal $\xi_n \equiv \xi_n(\boldsymbol{\alpha},\beta)$:

$$\xi_n(\boldsymbol{\alpha},\beta) = \beta\log\left(\frac{1-\rho_n(\boldsymbol{\alpha})}{\beta}\right)$$

- Plug-in

$$E_\beta(\boldsymbol{\alpha}) \;=\; \sum_j \alpha_j + \beta\frac{1}{N}\sum_{n=1}^{N}\log\left[1+\exp\left(\frac{1-\rho_n(\boldsymbol{\alpha})}{\beta}\right)\right] + \text{const.}$$

C-LP with Barriers III



- Barrier objective s.t. $\boldsymbol{\alpha} \geq \mathbf{0}$

$$E_\beta(\boldsymbol{\alpha}) \;=\; \sum_j \alpha_j + \beta\frac{1}{N}\sum_{n=1}^{N}\log\left[1+\exp\left(\frac{1-\rho_n(\boldsymbol{\alpha})}{\beta}\right)\right] + \text{const.}$$

- Has general form:

$$\min_{\boldsymbol{\alpha}\geq\mathbf{0}} G_\beta(\boldsymbol{\alpha}) + \boldsymbol{\kappa}^\top\boldsymbol{\alpha}$$

$\Rightarrow$ use general Leveraging Scheme to solve optimization problem

$\Rightarrow$ reduce barrier parameter during optimization

$\Rightarrow$ show convergence

Resulting Algorithm



1. **Input:** N examples $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$, T, C
2. **Initialize:** $f^{(0)} \equiv 0$, $\beta^{(1)} = \beta_{\text{start}}$, $d_n^{(1)} = [\exp(1/\beta^{(1)}) + 1]^{-1}$,
3. **Do for** $t = 1, \ldots, T$
   (a) Train weak learner on $\{S, \mathbf{d}^{(t)}\}$ and obtain $h_t : \mathbf{x} \mapsto [-1, 1]$
   (b) Let $\gamma_r = \sum_{n=1}^{N} y_n d_n^{(t)} h_r(\mathbf{x}_n)$ and $\underline{\alpha}_r = \sum_{s=1}^{t} \alpha_r \mathcal{I}(h_s = h_r)$ for $r = 1, \ldots, t$
   (c) $r^* = \operatorname{argmin}_{i \in \mathcal{I}} \gamma_s$, where $\mathcal{I} = \{i \mid i \in \{1, \ldots, t-1\} \text{ and } \underline{\alpha}_i > 0\}$.
   (d) **if** $\gamma_t - C < C - \tilde{\gamma}_{r^*}$ **then** $h_t = h_{r^*}$ and $\underline{\alpha}_t = \underline{\alpha}_{r^*}$ **else** $\underline{\alpha}_t = 0$
   (e) Find hypothesis coefficient by solving $\alpha_t = \operatorname{argmin}_{\alpha_t \geq -\underline{\alpha}_t} \mathrm{F}_{\beta^{(t)}}(\boldsymbol{\alpha}^{(t)})$
   (f) Let $f^{(t)} = f^{(t-1)} + \alpha_t h_t$.
   (g) Update weights $d_n^{(t+1)} = \dfrac{\exp([1 - y_n f^{(t)}(\mathbf{x}_n)]/\beta^{(t)})}{1 + \exp([1 - y_n f^{(t)}(\mathbf{x}_n)]/\beta^{(t)})}$.
   (h) **if** $\left|\sum_{n=1}^{N} d_n^{(t)} h_t(\mathbf{x}_n) - C\right| < \beta^{(t)}$, **then** $\beta^{(t+1)} := \texttt{next}(\beta^{(t)})$ **else** $\beta^{(t+1)} = \beta^{(t)}$

ν-LP-Regression



Input: - N examples $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$
- Hypotheses $h_1, \ldots, h_J$
- Regularization parameters C, ν

$$\text{Minimize } \frac{1}{N}\|\boldsymbol{\alpha}\|_1 + \frac{C}{N}\sum_{n=1}^{N}\xi_n + C\nu\epsilon \text{ such that}$$

$$* \; |f_{\boldsymbol{\alpha}}(\mathbf{x}_n) - y_n| \leq \epsilon + \xi_n$$
$$* \; \xi_n \geq 0$$
$$* \; \epsilon \geq 0$$

9. Applications of Boosting





Hill Inlet, Whitsunday Island

Main Sources:

- Schapire 2002
- Meir & Rätsch 2003

Practical Issues

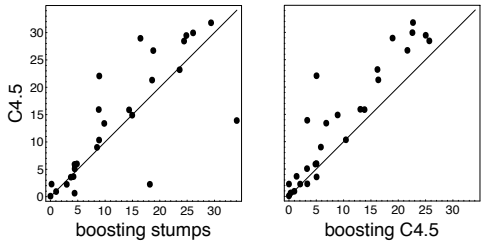


Advantages:	**Difficulties:**
A general meta-algorithm - use any 'reasonable' weak learner	Not clear how to incorporate prior knowledge effectively
Single parameter to be tuned (# iterations) - in principle	Regularization often essential - best strategy unclear
Fast and easy to program	The best choice of weak learner is not obvious
Theoretical performance guarantees	Decision boundaries generated using parallel-split based methods often very rugged

Performance on UCI Benchmarks



Weak Learners: Stumps and C4.5 decision trees



Source: Freund and Schapire 1996

Performance on Benchmarks



Weak Learners: Radial basis functions

	RBF	AB	AB_R	LP_R-AB	QP_R-AB	SVM
Banana	*10.8±0.6*	12.3±0.7	10.9±0.4	**10.7±0.4**	10.9±0.5	11.5±0.6
B.Cancer	27.6±4.7	30.4±4.7	26.5±5.5	26.8±6.1	**25.9±4.6**	*26.0±4.7*
Diabetes	24.1±1.9	26.5±2.3	*23.9±1.6*	24.1±1.9	25.4±2.2	**23.5±1.7**
German	24.7±2.4	27.5±2.5	*24.3±2.1*	24.8±2.2	25.2±2.1	**23.6±2.1**
Heart	17.1±3.3	20.3±3.4	16.6±3.7	**14.5±3.5**	17.2±3.4	*16.0±3.3*
Image	3.3±0.6	**2.7±0.7**	**2.7±0.6**	*2.8±0.6*	**2.7±0.6**	3.0±0.6
Ringnorm	*1.7±0.2*	1.9±0.3	**1.6±0.1**	2.2±0.5	1.9±0.2	*1.7±0.1*
F.Sonar	34.4±2.0	35.7±1.8	*34.2±2.2*	34.8±2.1	36.2±1.8	**32.4±1.8**
Splice	*9.9±1.0*	10.3±0.6	**9.5±0.7**	*9.9±1.4*	10.3±0.6	10.8±0.6
Thyroid	*4.5±2.1*	**4.4±2.2**	**4.4±2.1**	4.6±2.2	**4.4±2.2**	4.8±2.2
Titanic	23.3±1.3	*22.6±1.2*	*22.6±1.2*	24.0±4.4	22.7±1.1	**22.4±1.0**
Twonorm	*2.9±0.3*	3.0±0.3	**2.7±0.2**	3.2±0.4	3.0±0.3	3.0±0.2
Waveform	10.6±1.0	10.8±0.6	**9.8±0.8**	10.5±1.0	10.1±0.5	*9.9±0.4*
Mean %	6.6±5.8	11.9±7.9	1.7±1.9	8.9±10.8	5.8±5.5	4.6±5.4
Winner %	14.8±8.5	7.2±7.8	26.0±12.4	14.4±8.6	13.2±7.6	23.5±18.0

⋆ Highly competitive with state-of-the-art SVM classifier

Source: Rätsch *et al.* 2001

Other Applications



Some examples:

Text classification	Schapire and Singer - Used stumps with normalized term frequency and multi-class encoding
OCR	Schwenk and Bengio (neural networks)
Natural language Proc.	Collins; Haruno, Shirai and Ooyama
Image retrieval	Thieu and Viola
Medical diagnosis	Merle *et al.*
Fraud Detection	Rätsch & Müller 2001
Drug Discovery	Rätsch, Demiriz, Bennett 2002
Elect. Power Monitoring	Onoda, Rätsch & Müller 2000

Fuller list: Schapire's 2002, Meir & Rätsch 2003 review