Boosting and Ensemble Methods

K.-R. Müller[1,2] **G. Rätsch**[1,4] **S. Mika**[3]

[1] *FHG FIRST, Berlin and* [2] *University of Potsdam, Germany*

[3] *idalab GmbH, Berlin*

[4] *MPI Biological Cybernetics, Tübingen*

[1,2]**http://www.first.fhg.de/persons/Mueller.Klaus-Robert.html**

http://www.kernel-machines.org and **http://www.boosting.org**

Outline

- A brief Introduction to statistical Learning
- basic ideas: Boosting
- regularizing Adaboost
- mathematical programs: Boosting vs SVM
- conclusion

Basic ideas of statistical learning theory I

Three scenarios: regression, classification & density estimation.
Learn f from examples

$$(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N) \in \mathbf{R}^N \times \mathbf{R}^M \text{ or } \{\pm 1\}, \quad \text{generated from } P(\mathbf{x}, y),$$

such that expected number of errors on test set (drawn from $P(\mathbf{x}, y)$),

$$R[f] = \int \frac{1}{2} |f(\mathbf{x}) - y)|^2 \, dP(\mathbf{x}, y),$$

is minimal *(Risk Minimization (RM))*.

Problem: P is unknown. $\longrightarrow$ need an *induction principle*.

Empirical risk minimization (ERM): replace the average over $P(\mathbf{x}, y)$ by an average over the training sample, i.e. minimize the training error

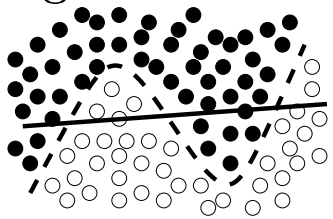
$$R_{emp}[f] = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{2} |f(\mathbf{x}_i) - y_i|^2$$

Basic ideas of statistical learning theory II

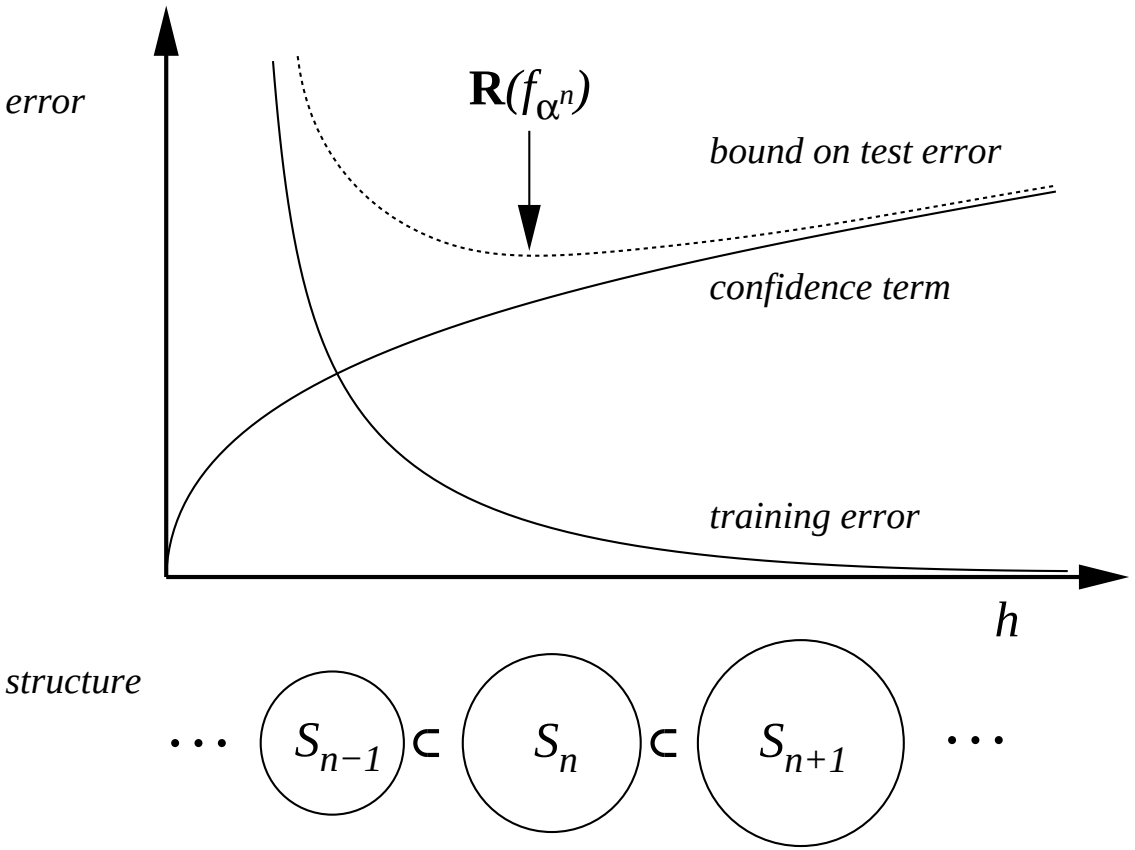
- Law of large numbers: $R_{emp}[f] \to R[f]$ as $N \to \infty$.
  "consistency" of ERM: for $N \to \infty$, ERM should lead to the same result as RM?
- No: *uniform* convergence needed (Vapnik) $\to$ VC theory.
  Thm. [classification] (Vapnik 95): with a probability of at least $1 - \eta$,

$$R[f] \leq R_{emp}[f] + \sqrt{\frac{d\left(\log \frac{2N}{d} + 1\right) - \log(\eta/4)}{N}}.$$

- Structural risk minimization (SRM): introduce structure on set of functions $\{f_\alpha\}$ & minimize RHS to get low risk! (Vapnik 95)
- d is VC dimension, measuring complexity of function class



SRM: the picture



Learning f requires small training error *and* small complexity of the set $\{f_\alpha\}$.

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

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

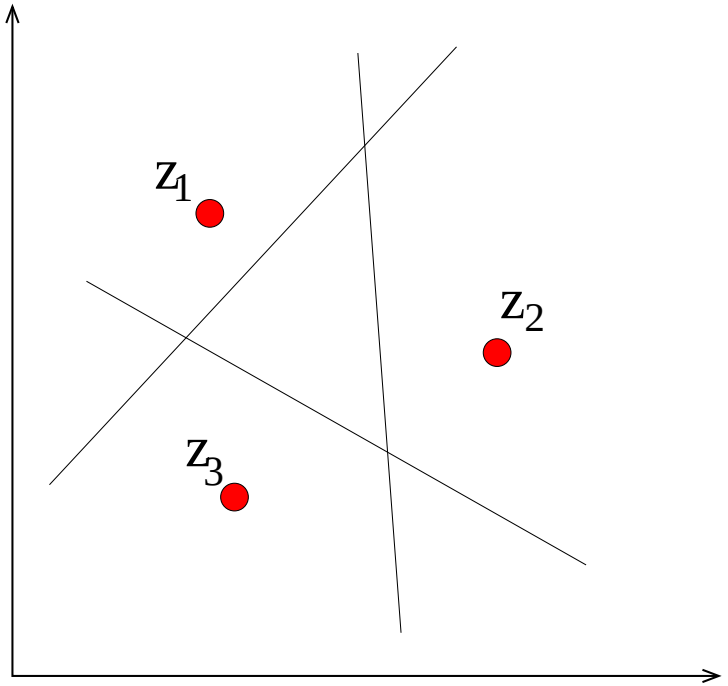
- **independent of the dimensionality of the space!**

VC-Dimension: Example

Half-spaces in $\mathbf{R}^2$:

$$f(x, y) = \text{sgn}(a + bx + cy), \quad \text{with parameters } a, b, c \in \mathbf{R}$$

- Clearly, we can shatter three non-collinear points.
- But we can never shatter four points.
- Hence the VC dimension is $d = 3$
- in n dimensions: VC dimension is $d = n + 1$

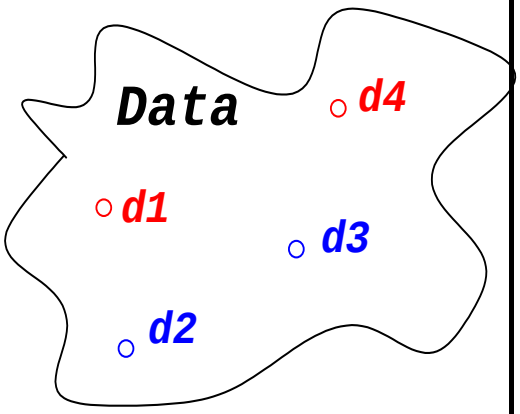


The basic idea behind Boosting

Ensemble learning for Classification

- Ensemble for binary classification consists of
  - Hypotheses (basis functions) $\{h_t(\mathbf{x}) : t = 1, \ldots, T\}$
    - of some hypothesis ("concept") set $\mathcal{H} = \{h \mid h(\mathbf{x}) \mapsto \{\pm 1\}\}$
  - Weights $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_T]$
    - satisfying $\alpha_t \geq 0$
- Classification Output: weighted majority of the votes
  - $f_{\text{Ens}}(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})$
- How to find the hypotheses and their weights?
  - Bagging (Breiman, 1996): $\alpha_t = 1/T$
  - AdaBoost (Freund & Schapire, 1994)

AdaBoost algorithm



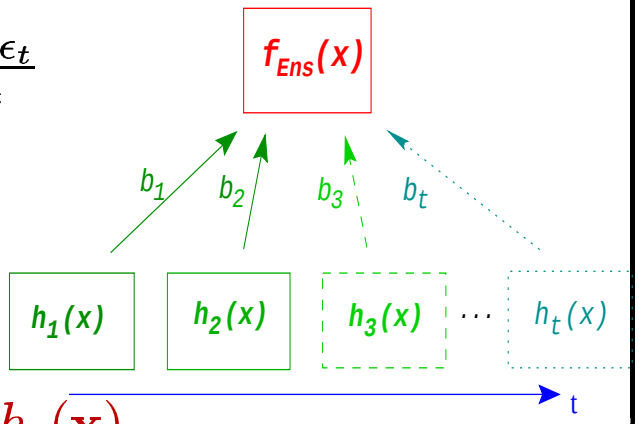
Input: N examples $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$

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

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

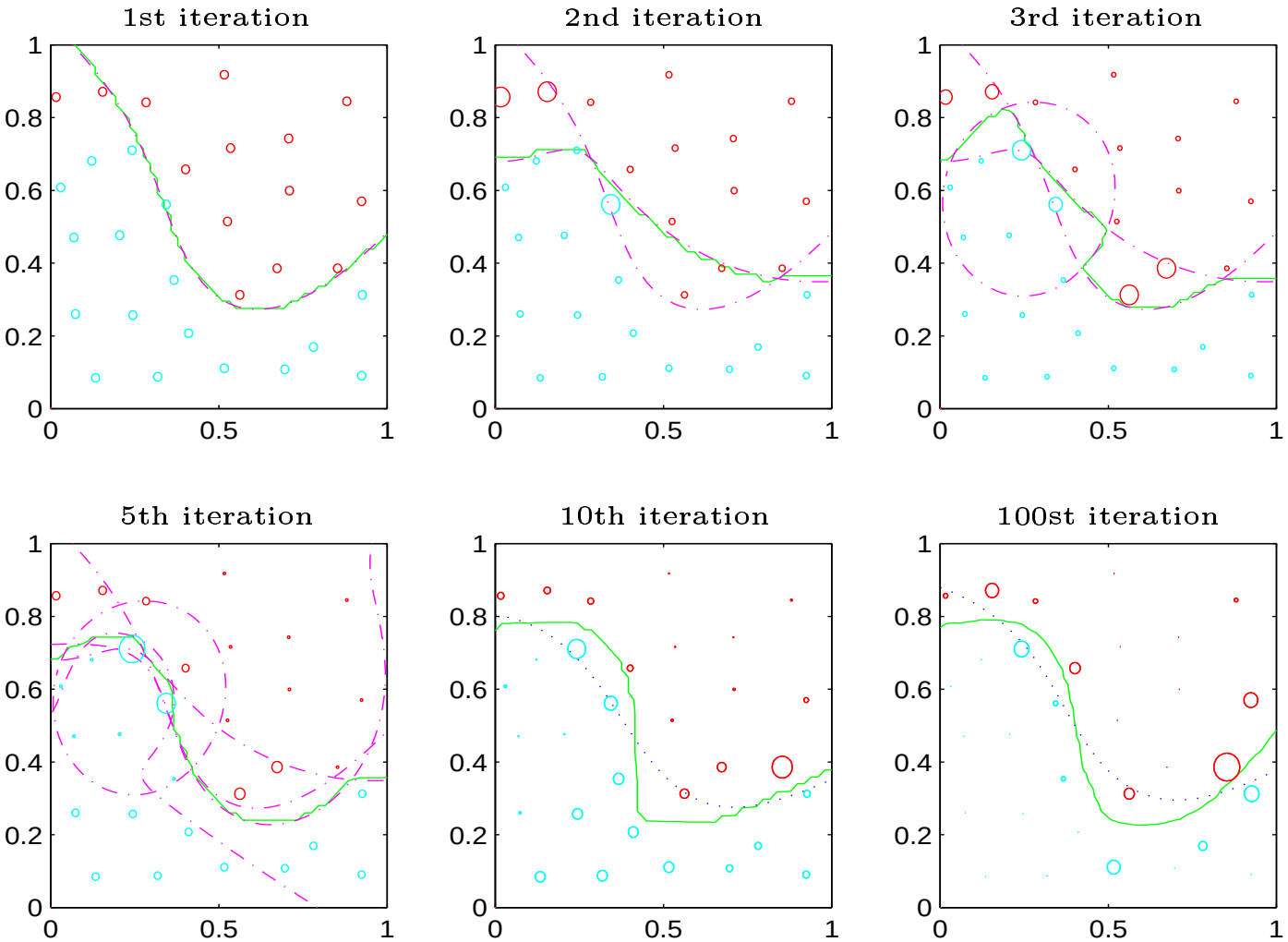
1. Train base learner according to example distribution $\mathbf{d}^{(t)}$ and obtain hypothesis $h_t : \mathbf{x} \mapsto \{\pm 1\}$.
2. compute weighted error $\epsilon_t = \sum_{i=1}^{N} d_i^{(t)} \mathrm{I}(y_i \neq h_t(\mathbf{x}_i))$
3. compute hypothesis weight $\alpha_t = \frac{1}{2} \log \frac{1-\epsilon_t}{\epsilon_t}$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

$$d_i^{(t+1)} = d_i^{(t)} \exp\left(-\alpha_t y_i h_t(\mathbf{x}_i)\right) / Z_t$$



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

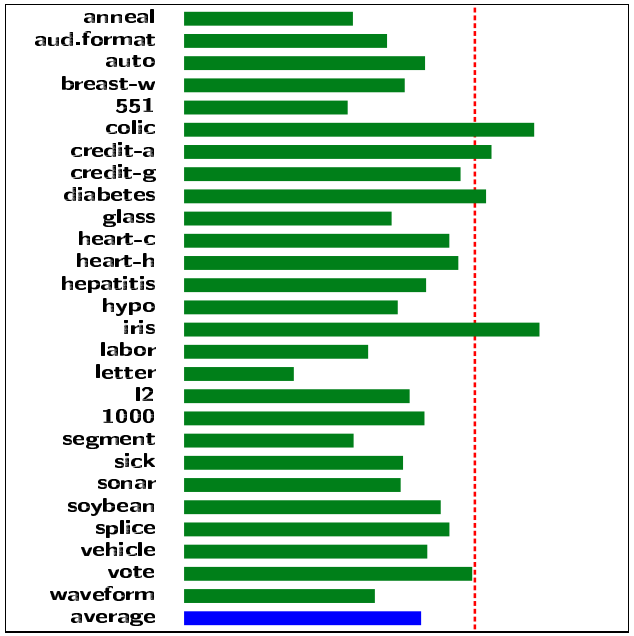
AdaBoost algorithm – An illustration



Motivation by Experiments

Architecture	Test Error
LeNet 1	1.7%
LeNet 4	1.1%
LeNet 5	0.9%
SVM polynom.	1.4%
SVM virt. SV	0.8%
boosted LeNet 4	**0.7%**

Comparison on NIST handwritten character recognition data set (LeCun et al. (1995))



Comparison on UCI repository data (Quinlan (1998))

Error Function for AdaBoost

- AdaBoost stepwise minimizes a function of $y_i f_{\boldsymbol{\alpha}}(x_i) = y_i \sum_t \alpha_t h_t(\mathbf{x}_i)$

$$\mathcal{G}(\boldsymbol{\alpha}) = \sum_{i=1}^{N} \exp\left\{-y_i f_{\boldsymbol{\alpha}}(\mathbf{x}_i)\right\}$$

- The gradient of $\mathcal{G}(\boldsymbol{\alpha}^{(t)})$ gives exactly the example weights used for AdaBoost:

$$\frac{\partial \mathcal{G}(\boldsymbol{\alpha}^{(t)})}{\partial f(\mathbf{x}_i)} \sim \exp\left\{-y_i f_{\boldsymbol{\alpha}}(\mathbf{x}_i)\right\} \sim d_i^{(t+1)}$$

- The hypothesis coefficient α_t is chosen, such that $\mathcal{G}(\boldsymbol{\alpha}^{(t)})$ is minimized:

$$\alpha_t = \underset{\alpha_t \geq 0}{\operatorname{argmin}}\, \mathcal{G}(\boldsymbol{\alpha}^{(t)}) = \frac{1}{2} \log \frac{1-\epsilon_t}{\epsilon_t}$$

- **AdaBoost is a gradient descent method to minimizes $\mathcal{G}(\boldsymbol{\alpha})$.**
  $\Longrightarrow$ Bregman Divergences (Entropy Projections, ...)
  $\Longrightarrow$ Coordinate Descent Methods & Column Generation

Theoretical Motivation – PAC Boosting

PAC Boosting – Exponential Convergence

Theorem 1 (Schapire et al. 1997) *Suppose AdaBoost generates hypotheses with weighted training errors* $\epsilon_1, \ldots, \epsilon_T$. *Then we have*

$$\sum_{i=1}^{N} \mathrm{I}(y_i \neq \mathrm{sign}(f_{Ens}(\mathbf{x}_i))) \leq 2^T \prod_{t=1}^{T} \sqrt{\epsilon_t(1-\epsilon_t)}$$

If $\epsilon_t < \frac{1}{2} - \frac{1}{2}\gamma$ (for all $t = 1, \ldots, T$), then the training error will decrease exponentially fast, i.e. will be **zero** after only

$$\frac{2\log(N)}{\gamma^2} = \mathcal{O}(\log(N))$$

iterations.

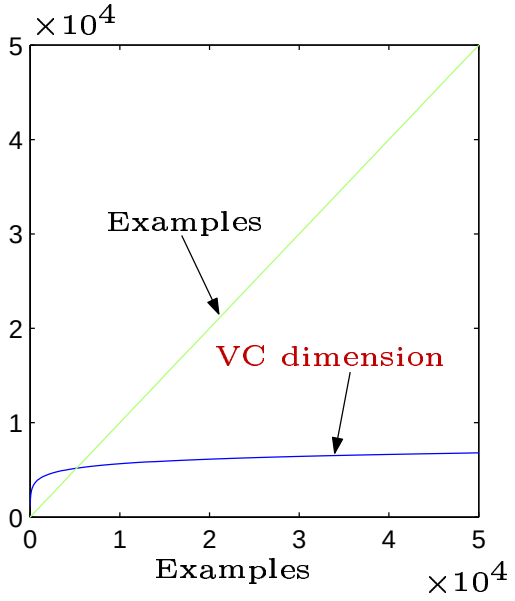
PAC Boosting – VC dim. of combined hypothesis

Let d be the VC dimension of the base hypothesis class $\mathcal{H}$.
Then the VC dimension of the class of combined functions is

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

An Example

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

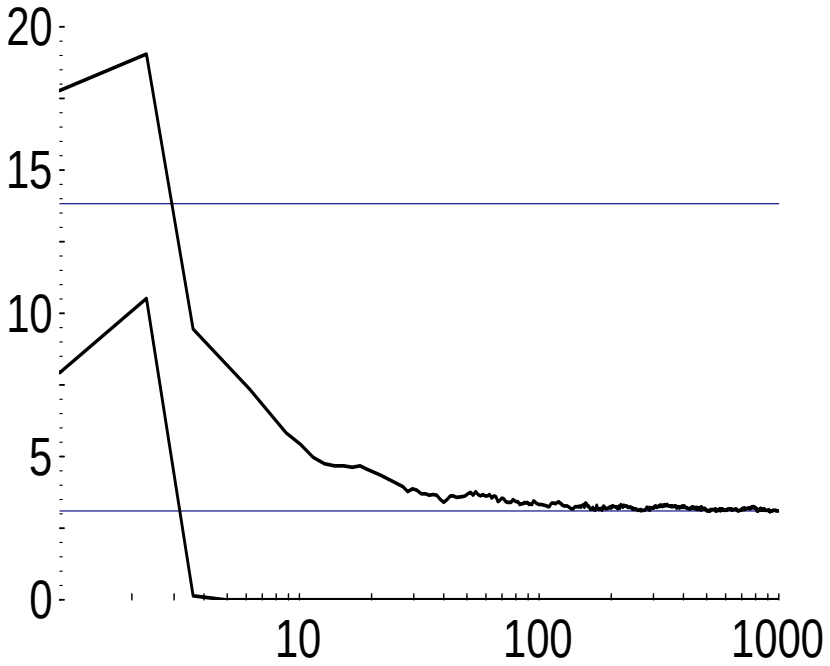


PAC Booosting – Digestion

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

A Strange Phenomenon

boosting C4.5 on "letter" data



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

Needs a better explanation!

Theoretical Motivation – Margin Distributions

Margin Distributions – Definitions

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

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

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

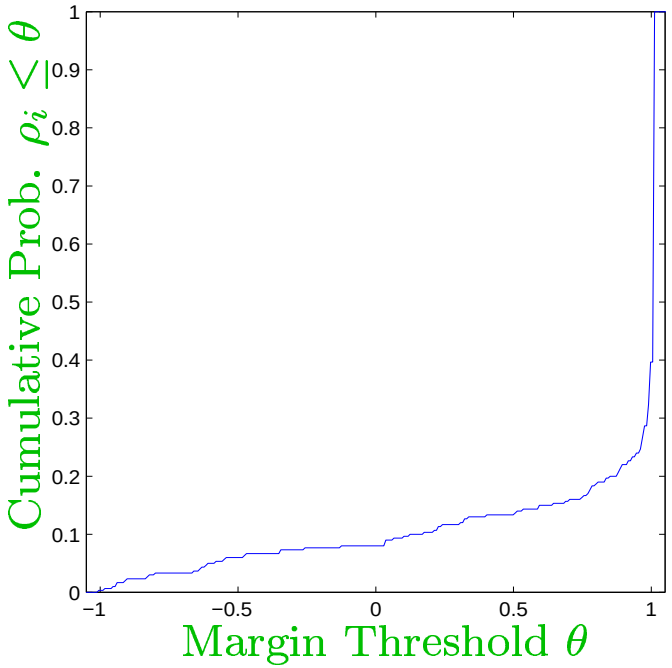
$$\rho_i(\boldsymbol{\alpha}) := y_i f_{\text{Ens}}(\mathbf{x}_i) = y_i \sum_{t=1}^{T} \frac{\alpha_t}{\sum_t \alpha_t} h_t(\mathbf{x}_i)$$

- Margin ϱ for a function f_{Ens} by $\varrho(\boldsymbol{\alpha}) := \min_{i=1,\ldots,N} \rho_i(\boldsymbol{\alpha})$

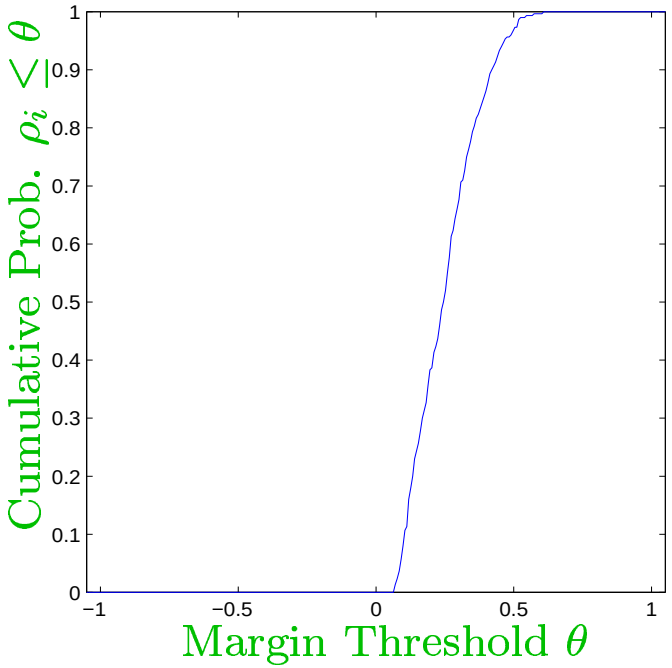
Margin Distributions – Illustration

AdaBoost tends to increase small margins, while decreasing large margins

Bagging



AdaBoost



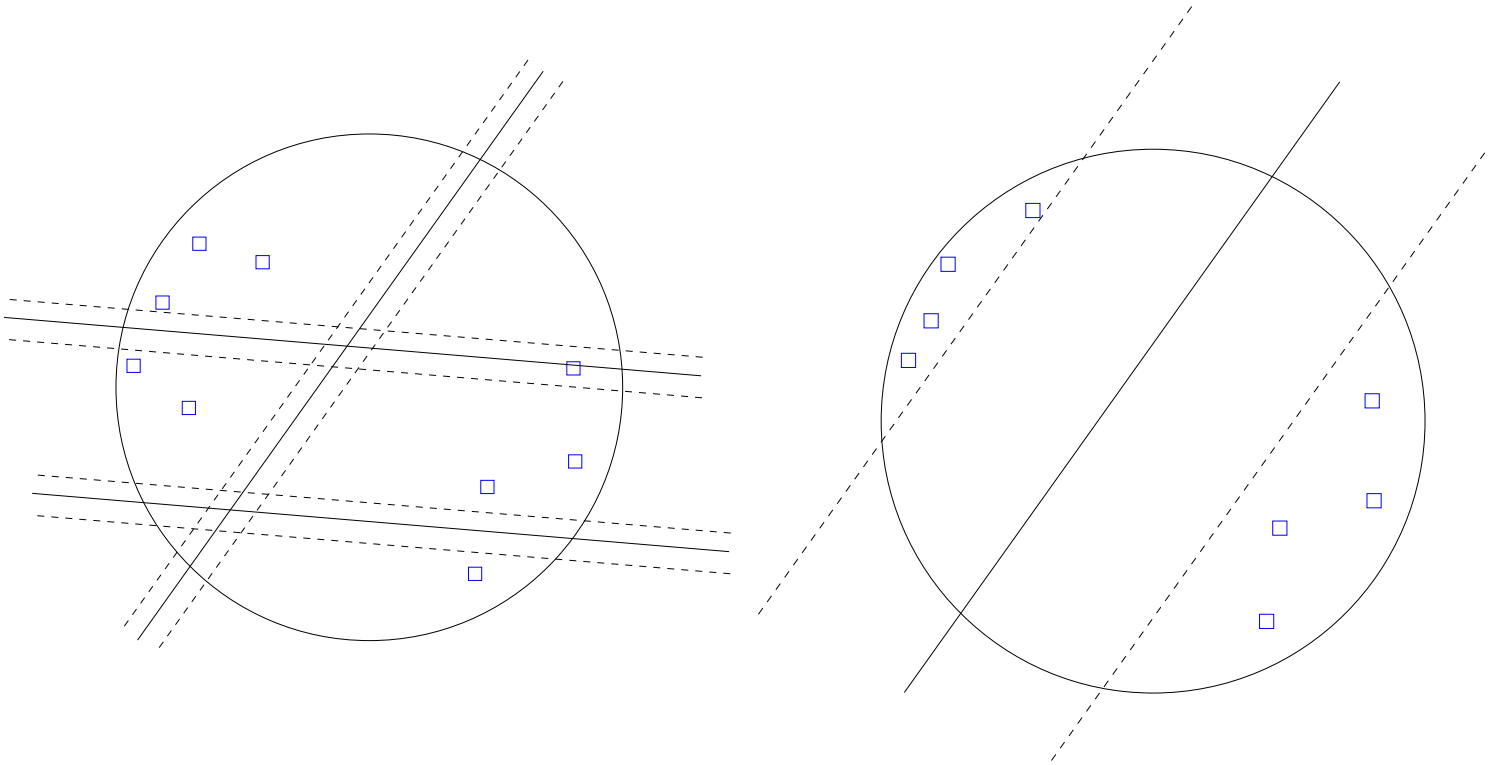
Margin Distributions – Lower Bounding the Margin

Theorem 2 *Suppose the base learning algorithm generates hypothesis with weighted training errors* $\epsilon_1, \ldots, \epsilon_T$. *Then we have for any* θ

$$P_{\mathbf{z}}(y f_{Ens}(\mathbf{x}) \leq \theta) \leq 2^T \prod_{t=1}^{T} \sqrt{{\epsilon_t}^{1-\theta}(1-\epsilon_t)^{1+\theta}}$$

Corollary 3 *If the base learning algorithm always achives* $\epsilon_t \leq \frac{1}{2} - \frac{1}{2}\gamma$ *then AdaBoost will generate a combined hyperplane with margin at least* $\frac{1}{2}\gamma$.

Margin Distributions – Large Margin Hyperplanes



Margin Distributions – A Bound

Theorem 4 *Let D be a distribution over $X \times \{\pm 1\}$ and let $\mathbf{Z}$ be a sample of N examples chosen independently at random according to D. Suppose the base-hypothesis space $\mathcal{H}$ has VC-dimension d, and let $\delta > 0$. Then with probability at least $1 - \delta$, the expected risk is bounded for $\theta > 0$ by*

$$R[f_{Ens}] \leq P_{\mathbf{Z}}(y f_{Ens}(\mathbf{x}) \leq \theta) + \mathcal{O}\left(\sqrt{\frac{d \log^2(N/d)}{N\theta^2} + \frac{\log(1/\delta)}{N}}\right)$$

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!**

Boosting in the Limit

Error Function for AdaBoost

- AdaBoost stepwise minimizes a function of $y_i f_{\boldsymbol{\alpha}}(x_i) = y_i \sum_t \alpha_t h_t(\mathbf{x}_i)$

$$\mathcal{G}(\boldsymbol{\alpha}) = \sum_{i=1}^{N} \exp\{-y_i f_{\boldsymbol{\alpha}}(\mathbf{x}_i)\}$$

- The gradient of $\mathcal{G}(\boldsymbol{\alpha}^{(t)})$ gives exactly the example weights used for AdaBoost:

$$\frac{\partial \mathcal{G}(\boldsymbol{\alpha}^{(t)})}{\partial f(\mathbf{x}_i)} \sim \exp\{-y_i f_{\boldsymbol{\alpha}}(\mathbf{x}_i)\} \sim d_i^{(t+1)}$$

- The hypothesis coefficient α_t is chosen, such that $\mathcal{G}(\boldsymbol{\alpha}^{(t)})$ is minimized:

$$\alpha_t = \underset{\alpha_t \geq 0}{\operatorname{argmin}}\, \mathcal{G}(\boldsymbol{\alpha}^{(t)}) = \frac{1}{2} \log \frac{1-\epsilon_t}{\epsilon_t}$$

- **AdaBoost is a coordinate gradient descent method which minimizes $\mathcal{G}(\boldsymbol{\alpha})$ stepwise.**

What happens in the long run?

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

$$d_i^{(t+1)} = \frac{\exp\left\{-\rho_i(\boldsymbol{\alpha}^{(t)})\right\}^{\|\boldsymbol{\alpha}^{(t)}\|}}{\sum_{j=1}^{N} \exp\left\{-\rho_j(\boldsymbol{\alpha}^{(t)})\right\}^{\|\boldsymbol{\alpha}^{(t)}\|}}$$

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

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

$\rightsquigarrow$ the d's concentrate on a **few** difficult patterns

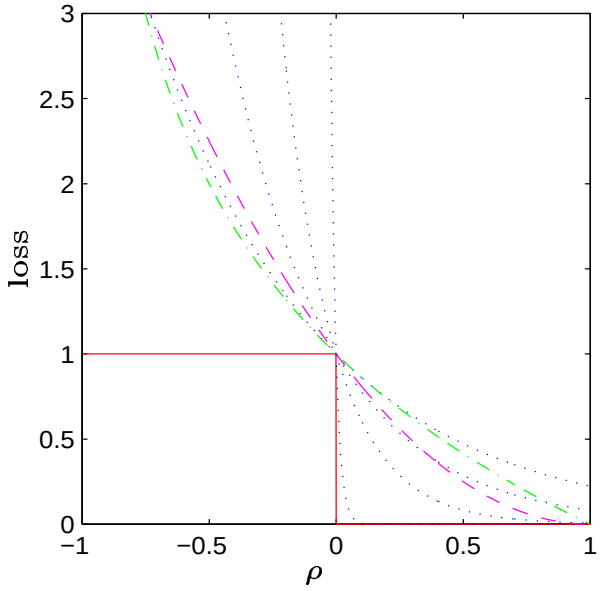
$\rightarrow$ **Support Patters**

$\rightsquigarrow$ Annealing Process:

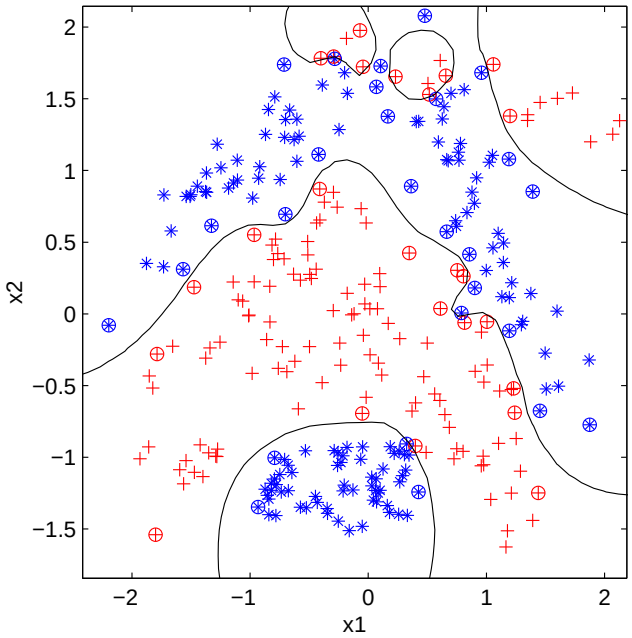
$$\mathcal{G}(\boldsymbol{\alpha}) = \sum \exp\left\{-\rho_i(\boldsymbol{\alpha})\right\}^{\|\boldsymbol{\alpha}\|_1}$$

$\rightarrow$ $0/\infty$-Loss approximated asymptoticaly

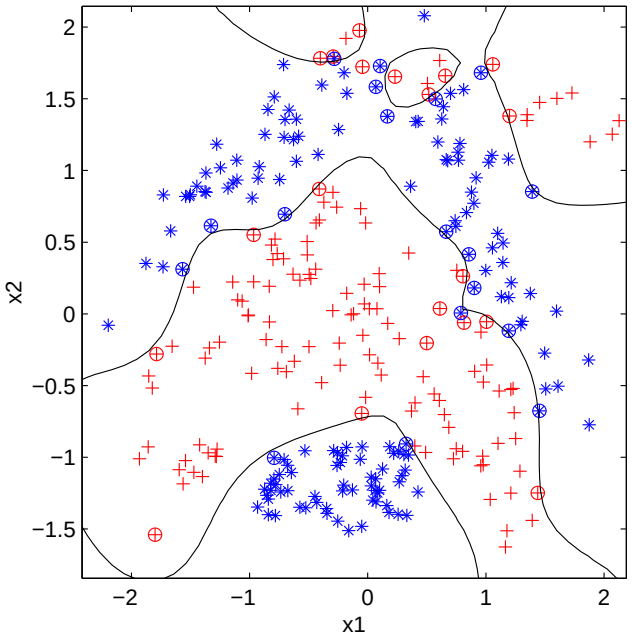
$\rightarrow$ **Barrier Optimization**



Support Patterns vs. Support Vectors



AdaBoost's decision line



SVM's decision line

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

Mathematical Programs: SVMs vs. Boosting

Mathematical Programming Formulation: SVM

The SVM minimization of

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

reformulate as maximization of the margin ρ

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

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

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

Boosting as mathematical program

- master hypothesis

$$f(\mathbf{x}) = \sum_{t=1}^{T} \frac{w_t}{\|\mathbf{w}\|_1} h_t(\mathbf{x})$$

- base hypotheses h_t produced by the base learning algorithm.

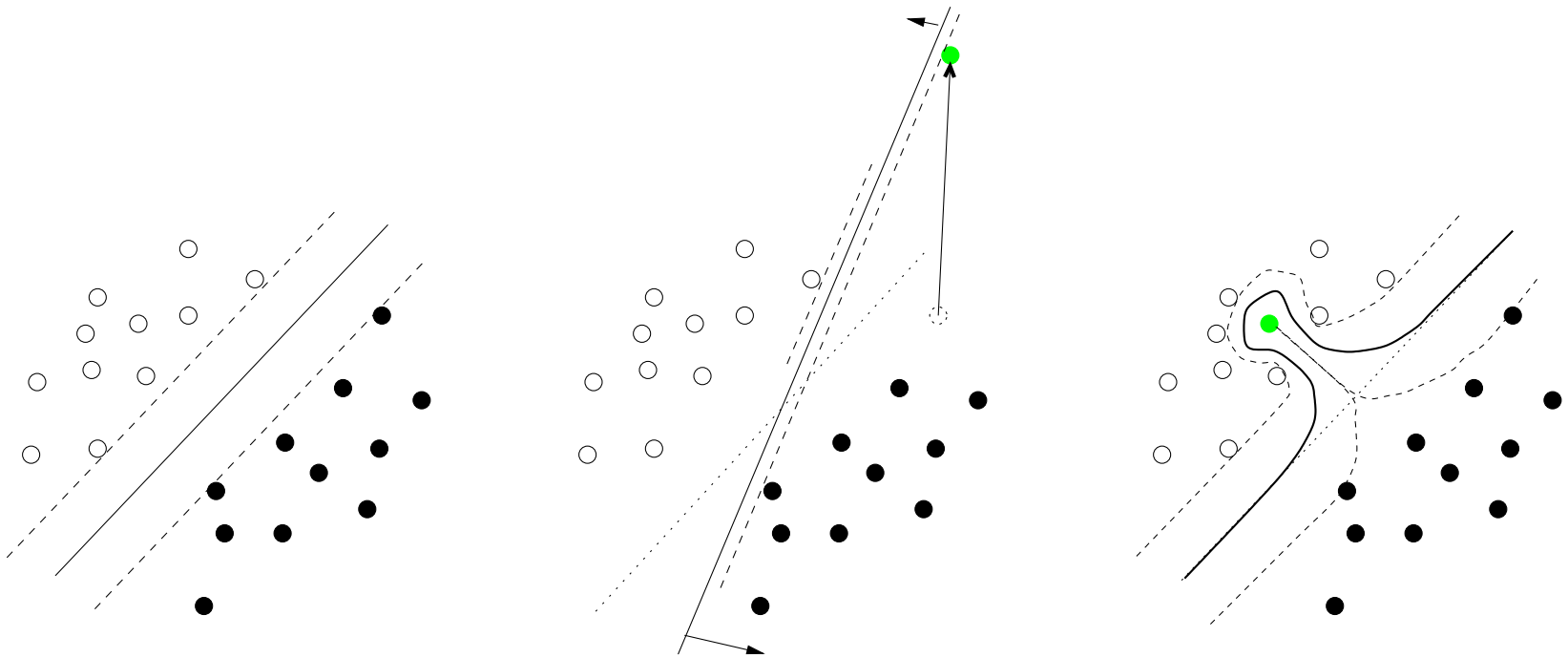
Arc-GV solution is asymptotically the same as linear program solution, maximizing smallest margin ρ:

$$\begin{aligned} \max_{\mathbf{w}\in\mathbf{R}^J, \rho\in\mathbf{R}_+} \quad & \rho \\ \text{subject to} \quad & y_i \sum_{j=1}^{J} w_j h_j(\mathbf{x}_i) \geq \rho \quad \text{for} \quad i = 1, \ldots, N \\ & \|\mathbf{w}\|_1 = 1, \end{aligned} \tag{3}$$

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

Soft Margins

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**.

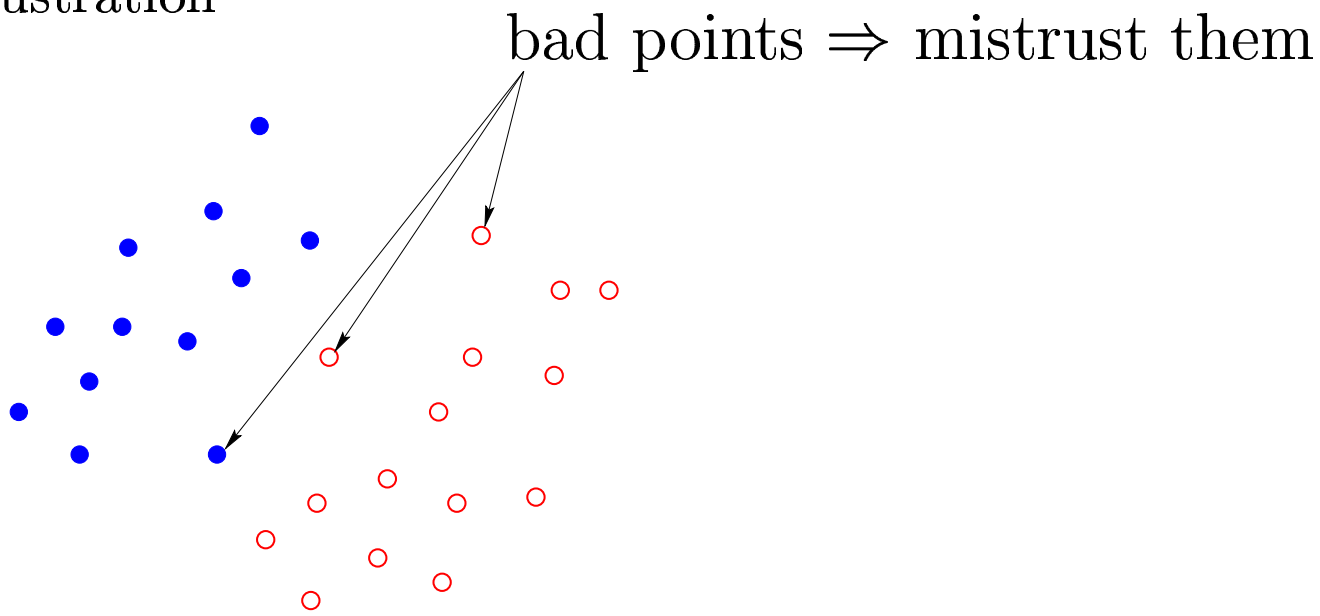
AdaBoost with Soft Margins

- 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



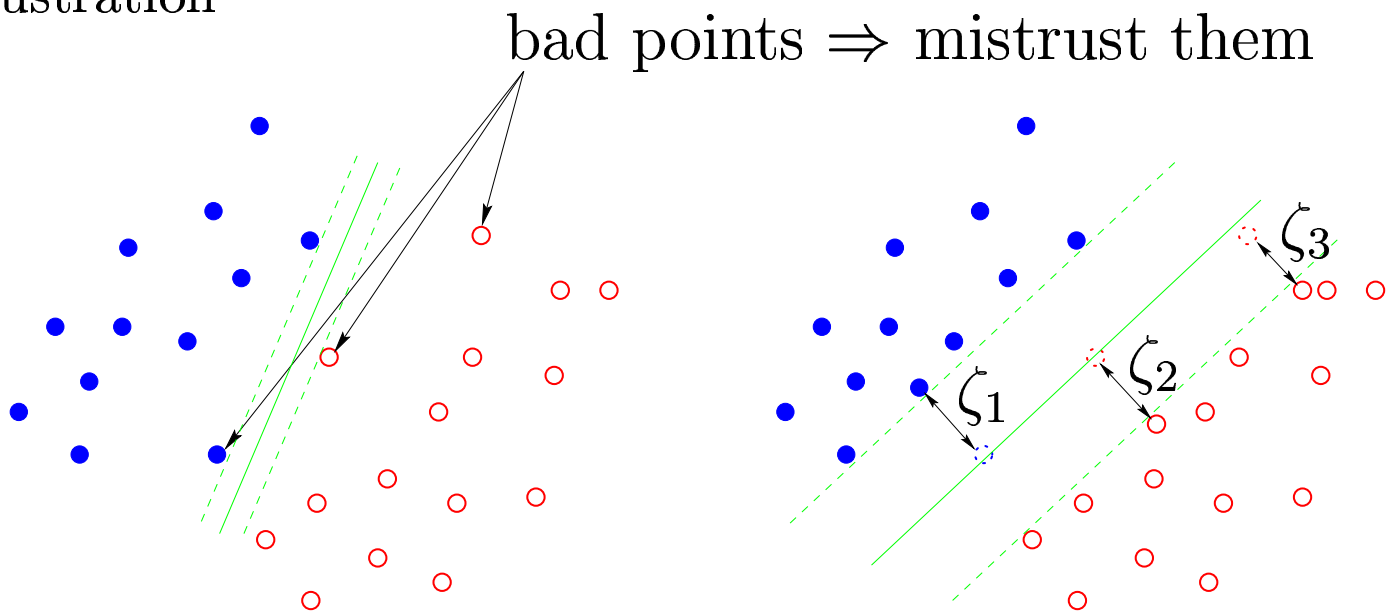
AdaBoost with Soft Margins

- 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 uncertainity measure ζ_n, we can easily get a new regularized Boosting algorithm.

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

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

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

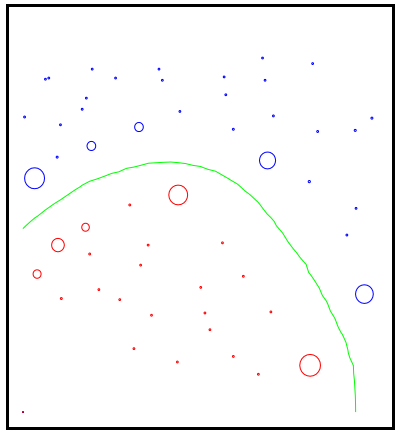
$$\alpha_t = \underset{\alpha_t \geq 0}{\operatorname{argmin}}\, \tilde{G}(\boldsymbol{\alpha})$$

AdaBoost$_{Reg}$ – Reducing the Influence

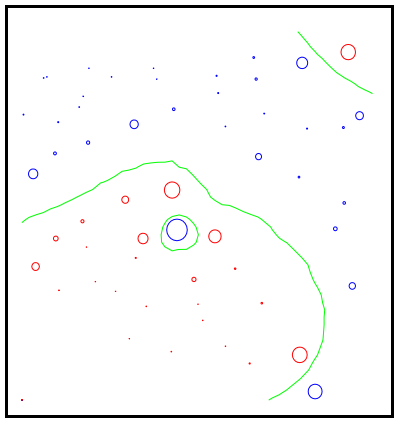
- How can we know which patterns are unreliable?

  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

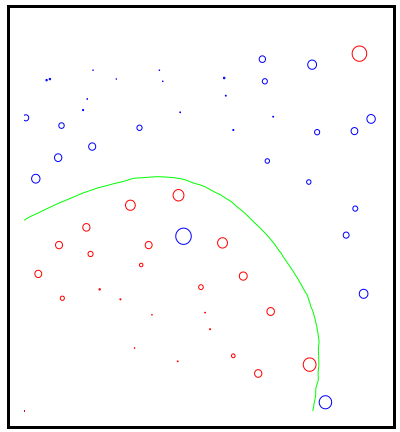
$$\mu_n^t = \sum_{r=1}^{t} \frac{\alpha_r}{\|\boldsymbol{\alpha}\|_1} d_n^r \qquad \zeta_n = C(\mu_n^t)^2$$



AdaBoost without "Noise"



AdaBoost with "Noise"



AdaBoost$_{Reg}$ with "Noise"

AdaBoost$_{Reg}$

Positive:

- first algorithm that addresses overfitting in Boostin
- 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

Benchmark Comparison

- 10 datasets (from UCI, DELVE and STATLOG repositories)
- Non binary problems partitioned into two-class problems.
- 100 partitions into test and training set (about 60%:40%).
- On each data sets we trained and tested all classifiers. Results are average test errors over 100 runs and standard deviations.
- Parameters estimated by 5-fold cross validation on first 5 realizations of dataset.
- For SVM we used Gaussian kernel
- For Boosting we used RBF networks as base learner

Experimental Results

	KNN	C4.5	RBF	AB	AB_R	SVM
Banana	15.0±1.0	16.1±2.8	10.8±0.6	12.3±0.7	10.9±0.4	11.5±0.7
B.Cancer	28.4±4.4	24.6±4.5	27.6±4.7	30.4±4.7	26.5±4.5	26.0±4.7
Diabetes	28.9±2.4	26.0±2.4	24.3±1.9	26.5±2.3	23.8±1.8	23.5±1.7
German	28.9±1.9	28.1±2.4	24.7±2.4	27.5±2.5	24.3±2.1	23.6±2.1
Heart	15.8±3.3	20.4±4.6	17.6±3.3	20.3±3.4	16.5±3.5	16.0±3.3
Ringnorm	35.9±1.3	15.3±1.5	1.7±0.2	1.9±0.3	1.6±0.1	1.7±0.1
F.Solar	37.8±2.8	33.2±1.9	34.4±2.0	35.7±1.8	34.2±2.2	32.4±1.8
Thyroid	5.8±2.8	8.7±3.3	4.5±2.1	4.4±2.2	4.6±2.2	4.8±2.2
Titanic	25.5±3.8	22.9±1.5	23.3±1.3	22.6±1.2	22.6±1.2	22.4±1.0
Waveform	11.4±0.8	17.8±1.0	10.7±1.1	10.8±0.6	9.8±0.8	9.9±0.4
Mean%	2400±6800	1200±2700	5.8±3.7	13.4±9.2	2.7±2.5	2.9±3.5

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 Processing	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

Conclusion

- Boosting algorithms
  - AdaBoost
  - PAC Motivation
  - Boosting with Large Margins
  - Strategies for Dealing with High Dimensional Spaces
  - Relations to Mathematical Programming & SVMs

`Boosting Homepage:` **http://www.boosting.org**

Sources of Information

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

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

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

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

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