Easy Learning Theory for Highly Scalable Algorithms

Claudio Gentile

DICOM

Universita' dell'Insubria, Italy

claudio.gentile@uninsubria.it

August, 2005

Content of this tutorial/1

Part 1

Intro to on-line learning problems, methods, relative loss bounds:

- On-line learning setting, examples
- Learning with expert advice (Bayes voting), relative loss bounds
- On-line learning linear-threshold functions
- Extensions: tracking the target, label-efficiency
- Learning ~~regression~~ functions

focus on BINARY classification

Content of this tutorial/2

Part 2

Intro to statistical Pattern Recognition problem, martingales, on-line to batch conversions:

- The statistical Pattern Recognition problem
- Types of error bounds (data/algorithm - independent/dependent)
- Reduction on-line pointwise $\rightarrow$ i.i.d.
  - Expectation analysis
  - Data-dependent analysis
- Final comments and conclusions

(Worst-case) on-Line Learning [L88,A88]

	experts						
	E_1	E_2	E_3	...	E_n	pred.	true lab.	loss
Day 1	1	1	-1	...	-1	-1	1	1
Day 2	1	-1	1	...	-1	1	-1	1
Day 3	-1	1	1	...	1	1	1	0
Day t	$z_{t,1}$	$z_{t,2}$	$z_{t,3}$	...	$z_{t,n}$	$\widehat{y}_t$	y_t	$\frac{1}{2}\lvert y_t - \widehat{y}_t\rvert$

On-line protocol

For $t = 1, ..., T$ do:

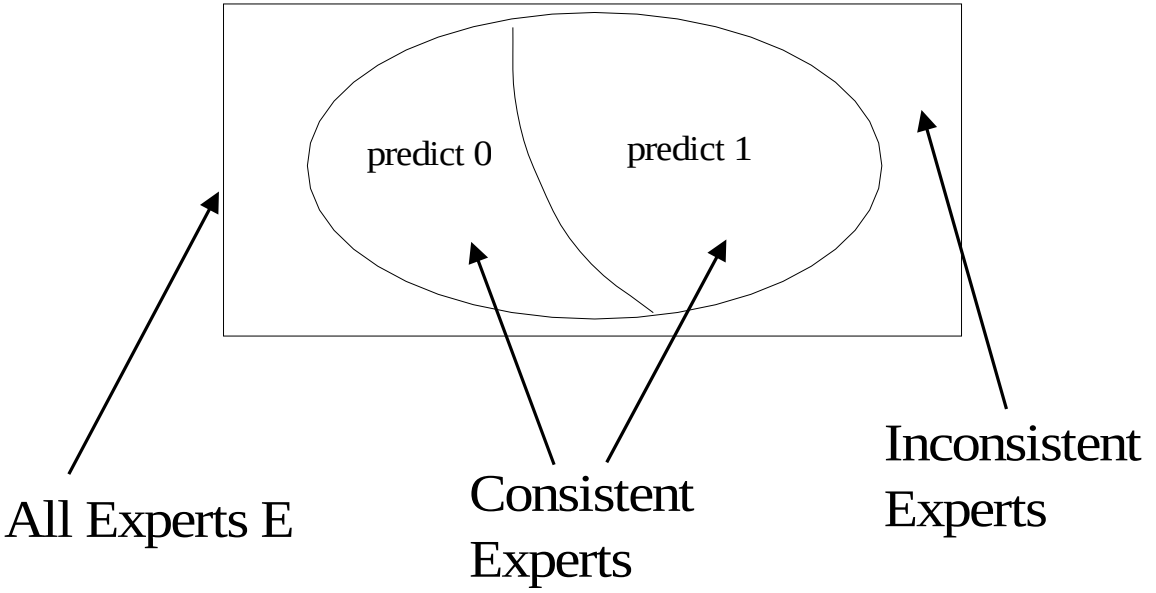
Get vector $\boldsymbol{z}_t \in \{-1, 1\}^n$

Predict $\widehat{y}_t \in \{-1, 1\}$

Get label $y_t \in \{-1, 1\}$

Incur loss $\frac{1}{2}\lvert y_t - \widehat{y}_t\rvert \in \{0, 1\}$

Halving Algorithm [BF72]



- Predicts with majority
- If mistake is made then number of consistent Experts is (at least) halved

A run of the Halving Algorithm (HA)

E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8	*majority*	*true label*	loss
1	1	-1	-1	1	1	-1	-1	1	-1	1
		-1	1			1	1	1	1	0
			1			-1	-1	-1	1	1
			↑							
			consistent							

$\forall$ sequence with k consistent experts (out of n)

HA makes $m \leq \log_2(n/k)$ mistakes: $n/2^m \geq k$

Learning with expert advice/1

What if no expert E_i is consistent?

Sequence of examples $S = (\boldsymbol{z}_1, y_1), \ldots, (\boldsymbol{z}_T, y_T)$

- $L_A(S)$ be total loss of alg. A on sequence S
- $L_i(S)$ be total loss of i-th expert E_i on S

Want bounds of the form:

$$\forall S: \quad L_A(S) \;\leq\; a \;\min_i L_i(S) + b \;\log(n)$$

where a, b are constants

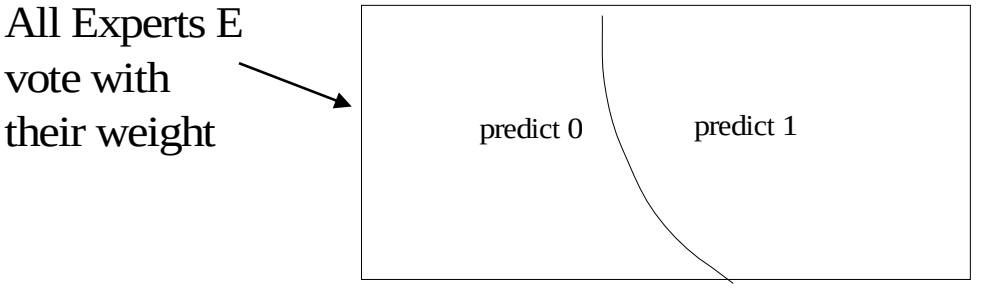
Bounds loss of algorithm relative to loss of best expert

Learning with expert advice/2

Can't wipe out experts!

Keep one weight per expert

The Weighted Majority Algorithm [L89a,LW94]



- Predicts with larger side
- Weights of wrong experts are slashed by a factor $\beta \in [0, 1)$

Learning with expert advice/3
Number of mistakes of the WM algorithm/1

$$
\begin{aligned}
L_{i,t} &= \#\text{ of mistakes of } E_i \text{ before time step } t \\
w_{t,i} &= \beta^{L_{i,t}} \text{ weight of } E_i \text{ at beginning of time step } t \\
W_t &= \sum_{i=1}^{n} w_{i,t} \quad \text{total weight at time step } t
\end{aligned}
$$

$$
\text{Minority} \leq \tfrac{1}{2} W_t \qquad \text{Majority} \geq \tfrac{1}{2} W_t
$$

If no mistake then minority multiplied by β: $W_{t+1} \leq 1\, W_t$

If mistake then majority multiplied by β:

$$
W_{t+1} \leq 1 \text{ Minority} + \beta \text{ Majority} \leq \frac{1+\beta}{2}\ W_t
$$

Learning with expert advice/3
Number of mistakes of the WM algorithm/2

Hence: $\underset{\text{total final weight}}{W_{T+1}} \leq \left(\frac{1+\beta}{2}\right)^{L_{WMA}(S)} W_1$

$$W_{T+1} = \sum_{j=1}^{n} w_{T+1,j} = \sum_{j=1}^{n} \beta^{L_j(S)} \geq \beta^{L_i(S)}$$

We got: $\left(\frac{1+\beta}{2}\right)^{L_{WMA}(S)} \underbrace{W_1}_{n} \geq \beta^{L_i(S)}$

Solving for $L_{WMA}(S)$: $L_{WMA}(S) \leq \frac{\ln 1/\beta}{\ln \frac{2}{1+\beta}} L_i(S) + \frac{1}{\ln \frac{2}{1+\beta}} \ln n$

$$L_{WMA}(S) \leq \underbrace{2.63}_{a} \min_i L_i(S) + \underbrace{2.63}_{b} \ln n$$

$\beta = 1/e$

Learning with expert advice/4

Slightly more general: absolute loss/1

Alg. A keeps exponential weights
and predicts with the weighted average [KW99]

$$
\begin{aligned}
w_{t,i} &= \beta^{L_{i,t}}, \quad L_{i,t} = \sum_{j<t} \tfrac{1}{2}|z_{i,j} - y_j| \\
v_{t,i} &= w_{t,i} / \textstyle\sum_{i=1}^{n} w_{t,i} \quad \text{normalized weights} \\
\widehat{y}_t &= \boldsymbol{v}_t^{\top} \boldsymbol{z}_t,
\end{aligned}
$$

where $z_{t,i} \in [-1, +1]$

Absolute loss of A: $L_A(S) = \sum_{t=1}^{T} \frac{1}{2}|\hat{y}_t - y_t|$

Learning with expert advice/4

Slightly more general: absolute loss/2

Relative loss bound

$\forall$ sequences $S = (\boldsymbol{z}_1, y_1), ..., (\boldsymbol{z}_T, y_T)$, $\boldsymbol{z}_t \in [-1, 1]^n$,
$y_t \in [-1, 1]$

$$L_A(S) \leq \min_i \underbrace{(1+\eta)}_{a} L_i(S) + \underbrace{\frac{1+\eta}{\eta}}_{b} \ln(n), \quad \eta = \log 1/\beta$$

Notice $a > 1$ (constant η)

Regret bounds ($a = 1$) need time-changing β [ACBG02]

Learning with expert advice/5

- Weighted Majority is just a Bayes voting scheme
- Easy to combine good experts (algorithms) so that prediction alg. is almost as good as best expert
- Bounds are logarithmic in # of experts

So far:

Learning relative to best expert/component

From now on:

Learning relative to best (thresholded) linear combination of experts/components

A more general setting

Instance	Prediction of alg A	Label	**Loss of alg A**
$\boldsymbol{x}_1$	$\widehat{y}_1$	y_1	$L(y_1, \widehat{y}_1)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\boldsymbol{x}_t$	$\widehat{y}_t$	y_t	$L(y_t, \widehat{y}_t)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\boldsymbol{x}_T$	$\widehat{y}_T$	y_T	$L(y_T, \widehat{y}_T)$
		Total Loss	$L_A(S)$

Sequence of examples $S = (\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_T, y_T) \in \mathbb{R}^n \times \{-1, 1\}$

Comparison class $\{\boldsymbol{u}\}$

Relative loss $\quad L_A(S) - \inf_{\{\boldsymbol{u}\}} Loss_{\boldsymbol{u}}(S)$

Goal: Bound relative loss for arbitrary sequence S

Learning linear-threshold functions/1
Another run of the Halving Algorithm/1

Sequence of examples $S = (\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_T, y_T) \in \mathbb{R}^2 \times \{-1, 1\}$

S is lin. separated by $\boldsymbol{u} \in \mathbb{R}^2 : ||\boldsymbol{u}||_2 = 1$ with margin

$0 < \gamma \leq y_t \boldsymbol{u}^\top \boldsymbol{x}_t \ \forall t$ $\qquad\qquad$ $R = \max_t ||\boldsymbol{x}_t||_2$

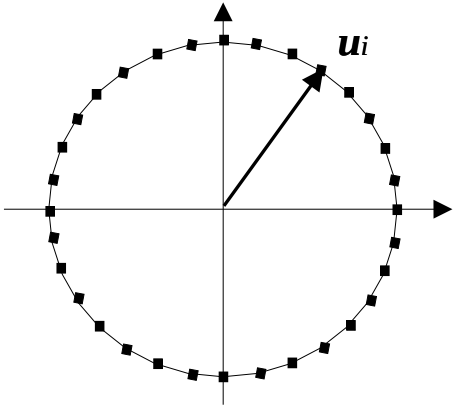
$\inf_{\{u\}}$ Loss $_u$ (S) = 0

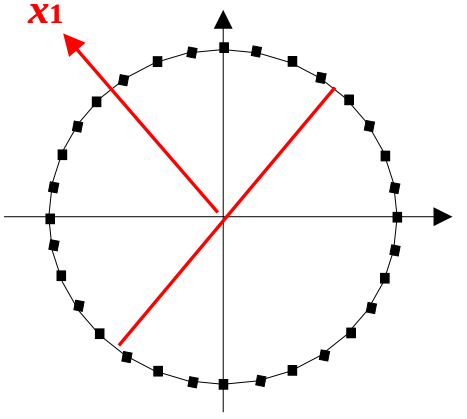
Experts:

n (large) linear−threshold functions evenly spread over unit circle

Expert i preditcts $z_{it} = \text{sgn}(\boldsymbol{u}_i^T \ \boldsymbol{x}_t)$

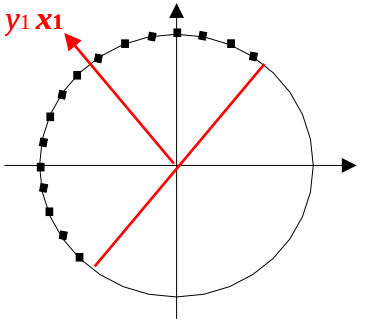
Feed experts with $\boldsymbol{x}_1$ and get expert prediction vector $\boldsymbol{z}_1$



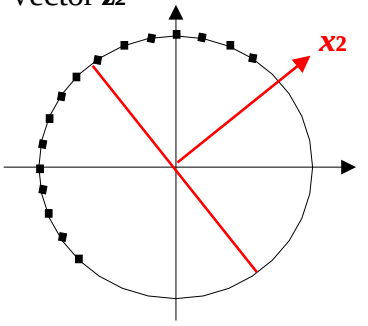


Learning linear-threshold functions/1
Another run of the Halving Algorithm/2

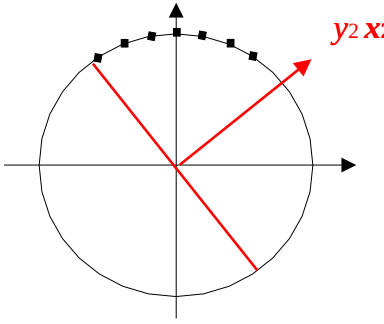
Get true label $y_1 = 1$ (mistake)
version space gets halved



Feed experts with x_2
and get expert prediction
vector z_2



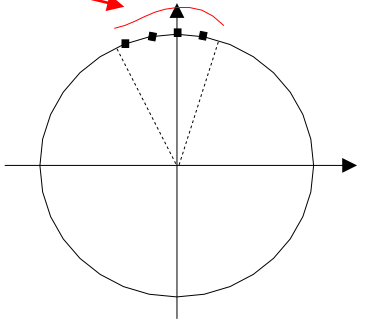
Get true label $y_2 = 1$ (mistake)
version space gets (at least) halved



...

...at the end

consistent experts ("margin")



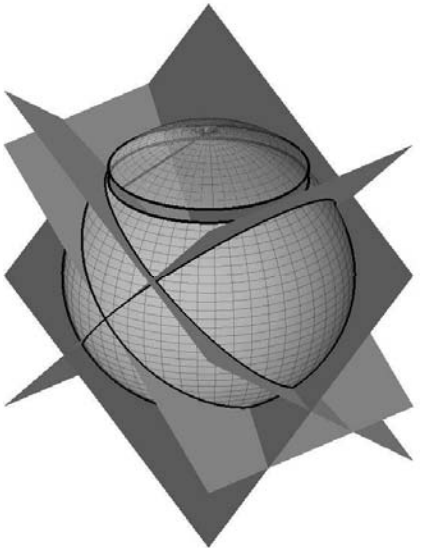
$$m_{HA} \leq \log_2(n/k) = O(\log(R/\gamma)) \text{ for large } n$$

Learning linear-threshold functions/1
Another run of the Halving Algorithm/3

[HG02,GBNT04,...]

For d-dim vectors:

$$m_{HA} \leq \log_2 1/\mu(\text{consistent}(S))$$
$$= O(d \log(R/\gamma)), \quad R = \max_t ||\boldsymbol{x}_t||_2$$



Courtesy: R. Herbrich

Proof: $y_t \boldsymbol{u}^\top \boldsymbol{x}_t \geq \gamma$ and $||\boldsymbol{u} - \boldsymbol{u}'||_2 < \gamma/R$

$\Longrightarrow$ $y_t (\boldsymbol{u}')^\top \boldsymbol{x}_t > 0$

$\Longrightarrow$ $\exists$ ball B of radius $\gamma/2R$: $B \subseteq \text{consistent}(S)$,

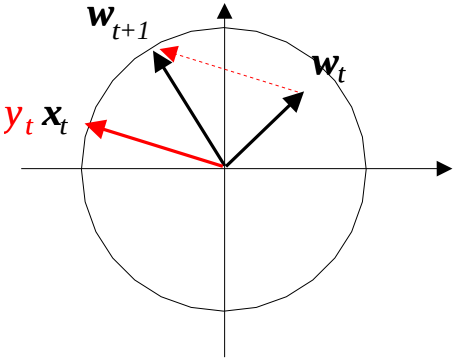
$\mu(B) = (\gamma/2R)^{d-1} \mu(\text{surface of } d\text{-dim unit sphere})$

Drawbacks: Linear dependence on dimension d

Looks too time-consuming (linear dependence on d?)

Learning linear-threshold functions/2

The (first-order) Perceptron algorithm [Ro62, ...]



Keep weight vector $\boldsymbol{w}_t \in \mathbb{R}^n$

In trial t:

- Get instance $\boldsymbol{x}_t \in \mathbb{R}^n$
- Predict with $\widehat{y}_t = \text{SGN}(\boldsymbol{w}_t^\top \boldsymbol{x}_t) \in \{-1, 1\}$
- Get label $y_t \in \{-1, 1\}$
- **If** mistake $(y_t \boldsymbol{w}_t^\top \boldsymbol{x}_t \leq 0)$ **then** update $\boldsymbol{w}_{t+1} := \boldsymbol{w}_t + y_t \, \boldsymbol{x}_t$

Learning linear-threshold functions/3

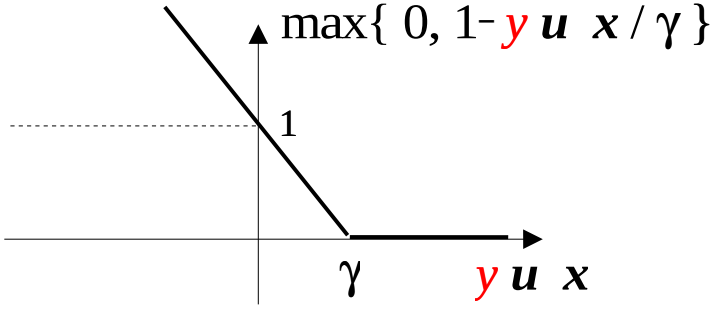
Perceptron convergence theorem/1 [Bl62,No62,...]

Arbitrary sequence $S = (\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_T, y_T) \in \mathbb{R}^n \times \{-1, 1\}$

$$\#\text{ of mistakes} \leq \inf_{\gamma>0,\ ||\boldsymbol{u}||_2=1} \left(\underbrace{D_\gamma(\boldsymbol{u}; S)}_{\text{"loss" of } \boldsymbol{u}} + \frac{\sqrt{\sum_{t\in\mathcal{M}} ||\boldsymbol{x}_t||_2^2}}{\gamma} \right),$$

$\mathcal{M}$ is set of mistaken trials t,

$D_\gamma(\boldsymbol{u}; S) = \sum_{t\in\mathcal{M}} \max\{0, 1 - y_t \boldsymbol{u}^\top \boldsymbol{x}_t/\gamma\}$



Learning linear-threshold functions/3

Perceptron convergence theorem/2

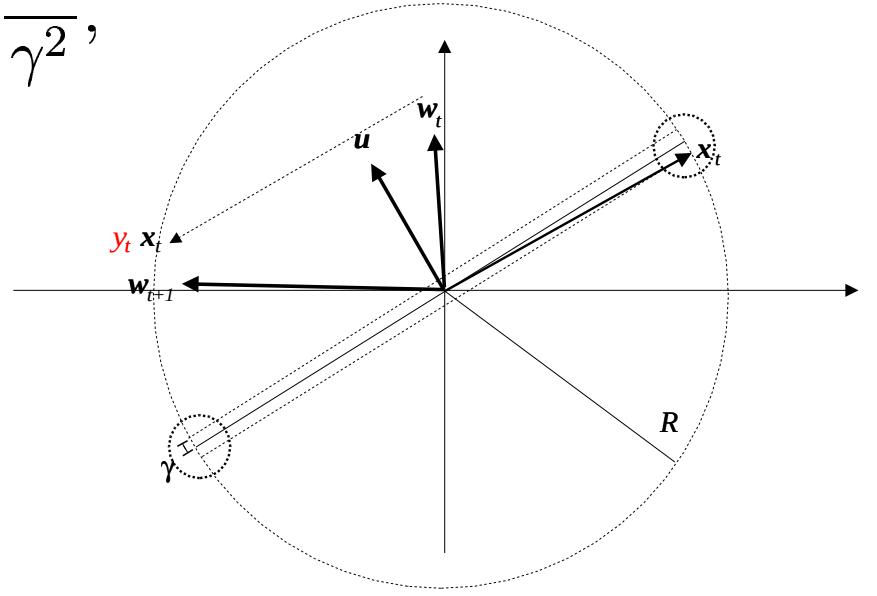
When S is separated by $\boldsymbol{u}$: $||\boldsymbol{u}||_2 = 1$ with margin

$\gamma \leq y_t \boldsymbol{u}^\top \boldsymbol{x}_t \ \forall t$

gets

$$\# \text{ of mistakes} \leq \frac{R^2}{\gamma^2},$$

$||\boldsymbol{x}_t|| \leq R$



Pointwise bound:

Depends on radius R and margin γ

Learning linear-threshold functions/4

The second-order Perceptron algorithm [CBCG05]

Keep weight vector $\boldsymbol{w}_t \in \mathbb{R}^n$ and matrix S_t

In trial t:

positive parameter

- Get instance $\boldsymbol{x}_t \in \mathbb{R}^n$
- Predict with $\widehat{y}_t = \text{SGN}(\boldsymbol{w}_t^\top (aI + S_t)^{-1} \boldsymbol{x}_t) \in \{-1, 1\}$
- Get label $y_t \in \{-1, 1\}$
- **If** mistake **then** update
  - $\boldsymbol{w}_{t+1} := \boldsymbol{w}_t + y_t \hat{\boldsymbol{x}}_t$
  - $S_{t+1} = S_t + \hat{\boldsymbol{x}}_t \hat{\boldsymbol{x}}_t^\top$, $\qquad \hat{\boldsymbol{x}}_t = \boldsymbol{x}_t / ||\boldsymbol{x}_t||$

Turns to first-order when $a \to \infty$

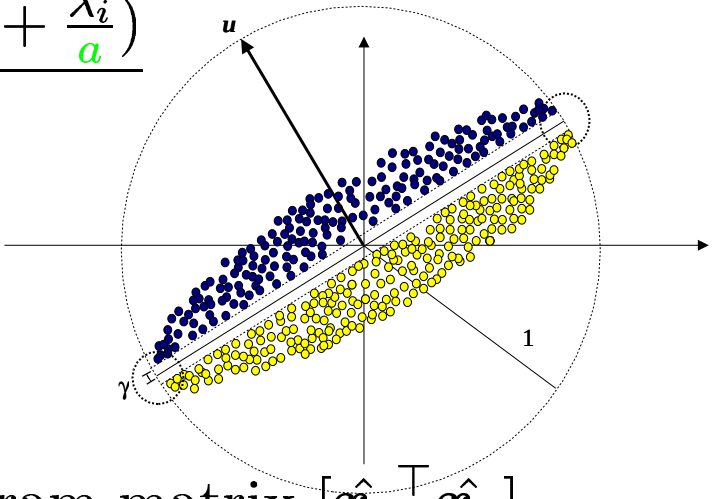
Learning linear-threshold functions/5

Second-order convergence theorem [Ge04]

When $S = (\hat{\boldsymbol{x}}_1, y_1), ..., (\hat{\boldsymbol{x}}_T, y_T) \in \mathbb{R}^n \times \{-1, 1\}$
is separated by $\boldsymbol{u}$ with margin $\gamma \leq y_t \boldsymbol{u}^\top \hat{\boldsymbol{x}}_t,\ ||\hat{\boldsymbol{x}}_t|| \leq 1 \, \forall t$
gets

$$\#\text{ of mistakes} \leq \frac{a + \sum_{i=1}^{n} \ln(1 + \frac{\lambda_i}{a})}{\gamma}$$

More complicated bound
in the nonseparable case



Pointwise bound:
Depends on eigenstructure $\{\lambda_i\}$ of Gram matrix $[\hat{\boldsymbol{x}}_j{}^\top \hat{\boldsymbol{x}}_k]_{j,k \in \mathcal{M}}$
and linearly on inverse margin γ

Learning linear-threshold functions/6

Kernel Perceptron [FS98,...]

Keep pool of "support vectors" $\mathcal{M}_t$

In trial t:

- Get instance $\boldsymbol{x}_t \in \mathbb{R}^n$
- Predict with $\widehat{y}_t = \text{SGN}(\sum_{i \in \mathcal{M}_t} y_i \, K(\boldsymbol{x}_i, \boldsymbol{x}_t)) \in \{-1, 1\}$
- Get label $y_t \in \{-1, 1\}$
- **If** mistake **then** update $\mathcal{M}_{t+1} := M_t \cup \{t\}$

Learning linear-threshold functions/7
Kernel Perceptron convergence theorem/1

Arbitrary sequence $S = (\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_T, y_T) \in \mathbb{R}^n \times \{-1, 1\}$

$$\#\text{ of mist.} \leq \inf_{\gamma>0,\ f \in H_K, \|f\|=1} \left(\underbrace{D_\gamma(f;S)}_{\text{"loss" of } f} + \frac{\sqrt{\sum_{t \in \mathcal{M}} K(\boldsymbol{x}_t, \boldsymbol{x}_t)}}{\gamma} \right)$$

$H_K = \{f(\cdot) = \sum_{t=1}^{T} \alpha_t K(\boldsymbol{x}_t, \cdot) \ : \ \alpha_t \in \mathbb{R}\}$,

$\mathcal{M}$ is set of mistaken trials t,

$D_\gamma(f;S) = \sum_{t \in \mathcal{M}} \max\{0, 1 - y_t f(\boldsymbol{x}_t)/\gamma\}$

Separable case:

$$\#\text{ of mistakes} \leq R^2/\gamma^2, \quad K(\boldsymbol{x}_t, \boldsymbol{x}_t) \leq R^2$$

Learning linear-threshold functions/8

Kernel Second-order Perceptron [CBCG05]

Keep pool of "support vectors" $\mathcal{M}_t$

In trial t:

- Get instance $\boldsymbol{x}_t \in \mathbb{R}^n$
- Predict with $\widehat{y}_t =$

$$\text{SGN}\left(\boldsymbol{y}_t^\top \left(a\,I + \underbrace{[\hat{K}(\boldsymbol{x}_i, \boldsymbol{x}_j)]_{i,j\in\mathcal{M}_t}}_{current\ Gram\ matrix}\right)^{-1} \boldsymbol{v}_t\right) \in \{-1, 1\},$$

$\boldsymbol{y}_t$: y_i, $i \in M_t$

$\boldsymbol{v}_t$: $\hat{K}(x_i, x_t)$, $i \in M$

- Get label $y_t \in \{-1, 1\}$
- **If** mistake **then** update $\mathcal{M}_{t+1} := M_t \cup \{t\}$

Learning linear-threshold functions/9

Kernel Second-order convergence theorem

When $S = (\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_T, y_T) \in \mathbb{R}^n \times \{-1, 1\}$
is separated by $f(\cdot) = \sum_{t=1}^{T} \alpha_t \hat{K}(\boldsymbol{x}_t, \cdot)$, $\alpha_t \in \mathbb{R}$,
with margin $\gamma \leq y_t f(\boldsymbol{x}_t)$ $\forall t$

gets

$$\# \text{ of mist.} \leq \frac{a + \sum_i \ln(1 + \frac{\lambda_i}{a})}{\gamma},$$

λ_i is i-th eigenvalue of (normalized) kernel Gram matrix
$[\hat{K}(\boldsymbol{x}_i, \boldsymbol{x}_j)]_{i,j \in \mathcal{M}}$,

$\mathcal{M}$ is set of mistaken trials

Learning linear-threshold functions/10

Second-order Perceptron: computational aspects

Time per trial

Primal formulation

compute $(aI + S_t)^{-1}$ based on $(aI + S_{t-1})^{-1}$

$O(n^2)$ extra time per trial

Dual formulation

compute $\left(aI + [\hat{G}(\boldsymbol{x}_i, \boldsymbol{x}_j)]_{i,j \in \mathcal{M}_t}\right)^{-1}$

based on $\left(aI + [\hat{G}(\boldsymbol{x}_i, \boldsymbol{x}_j)]_{i,j \in \mathcal{M}_{t-1}}\right)^{-1}$

$O(|\mathcal{M}_t|^2)$ extra inner products (kernel evaluations) per trial

# of mistakes so far

Learning linear-threshold functions/11
Additive algorithms

An additive algorithm (e.g. first/second-order Perceptron):

- Relies on linear algebra
- Is rotation invariant (depends on data via angles)
- Can be easily kernelized ($\boldsymbol{x}_i^\top \boldsymbol{x}_j \to K(\boldsymbol{x}_i, \boldsymbol{x}_j)$)
- Has no bias for axes-parallel directions (no feature selection)

Learning linear-threshold functions/12
Nonadditive algorithms

- No linear algebra
- No rotation invariance
- Harder to kernelize
- Bias for sparse solutions (built-in feature selection)

Example: p-norm algorithms

Learning linear-threshold functions/13

p-norm algs [GLS01,GL99,Ge03]

Keep weight vector $\boldsymbol{w}_t \in \mathbb{R}^n$

In trial t:

$$\boldsymbol{f}(\cdot) = \nabla \tfrac{1}{2}||\cdot||_p^2,\ p \geq 2$$

- Get instance $\boldsymbol{x}_t \in \mathbb{R}^n$
- Predict $\widehat{y}_t = \text{SGN}(\boldsymbol{f}(\boldsymbol{w}_t)^\top \boldsymbol{x}_t) \in \{-1, 1\}$
- Get label $y_t \in \{-1, 1\}$
- **If** mistake **then** update $\boldsymbol{w}_{t+1} := \boldsymbol{w}_t + y_t\, \boldsymbol{x}_t$

Notice:

- $p = 2$ gets (first-order) Perceptron
- $p = O(\ln n)$ gets Weighted Majority/Winnow [L88,LW94]
- $2 < p < O(\ln n)$ interpolates between the two extremes

Learning linear-threshold functions/14

p-norm Perceptron convergence theorem/1

[GLS01,GL99,Ge03]

Arbitrary sequence $S = (\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_T, y_T) \in \mathbb{R}^n \times \{-1, 1\}$

$$\# \text{ mistakes} \leq \inf_{\gamma>0,\ ||\boldsymbol{u}||_q=1} \left(\underbrace{D_\gamma(\boldsymbol{u}; S)}_{\text{"loss" of } \boldsymbol{u}} + \frac{\sqrt{(p-1)\sum_{t\in\mathcal{M}} ||\boldsymbol{x}_t||_p^2}}{\gamma} \right),$$

$\mathcal{M}$ is set of mistaken trials t,

$D_\gamma(\boldsymbol{u}; S) = \sum_{t\in\mathcal{M}} \max\{0, 1 - y_t \boldsymbol{u}^\top \boldsymbol{x}_t/\gamma\}$

Learning linear-threshold functions/14

p-norm Perceptron convergence theorem/2

When S is separated by $\boldsymbol{u}$: $||\boldsymbol{u}||_q = 1$ with margin

$\gamma \leq y_t \boldsymbol{u}^\top \boldsymbol{x}_t \ \forall t \quad \underbrace{(1/p + 1/q = 1)}_{\text{dual norms}}$

gets

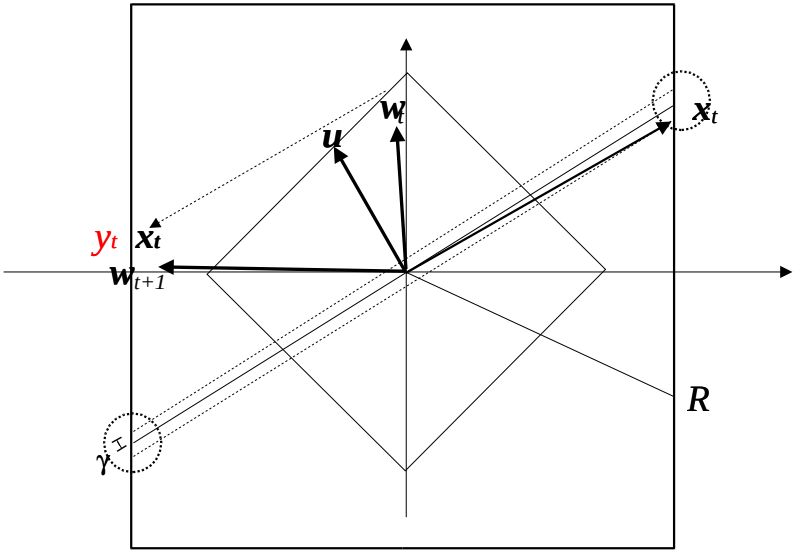
$$\#\text{ of mistakes} \leq (p-1)\frac{R^2}{\gamma^2}$$

$||\boldsymbol{x}_t|| \leq R$

Pointwise bound:

Depends on p-norm radius R

and (q-norm) margin γ



Learning linear-threshold functions/15

p-norm algorithms with kernels/1 (wild slide) [Ge04]

$$\boldsymbol{x} = \begin{matrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{matrix} \quad \Rightarrow \quad \Phi(\boldsymbol{x}) = \begin{matrix} 1 \\ x_1 \\ x_2 \\ \vdots \\ x_n \\ x_1\,x_2 \\ \vdots \\ x_1\,x_2\,\ldots\,x_n \end{matrix}$$

$$K(\boldsymbol{x},\boldsymbol{y}) = \Phi(\boldsymbol{x})^\top \Phi(\boldsymbol{y}) = \prod_{i=1}^{n} (1 + x_i\,y_i)$$

(Simple poly kernel)

Learning linear-threshold functions/15

p-norm algorithms with kernels/2 (wild slide)

p-norm hypothesis: $\boldsymbol{w} = \sum_{i \in \mathcal{M}} y_i \Phi(\boldsymbol{x}_i)$

p-norm margin: $= \boldsymbol{f}(\boldsymbol{w})^\top \Phi(\boldsymbol{x})$ $\qquad\qquad$ $\boldsymbol{f}(\boldsymbol{w}) = \boldsymbol{w}^{p-1}$

$$= \underbrace{\left(\sum_{i \in \mathcal{M}} y_i \Phi(\boldsymbol{x}_i)^\top \right)^{p-1}}_{expand!} \Phi(\boldsymbol{x})$$

Then expand polynomial and use $\Phi(\boldsymbol{x})\Phi(\boldsymbol{y}) = \Phi(\boldsymbol{xy})$

Learning linear-threshold functions/15

p-norm algorithms with kernels/3 (wild slide)

Example: $p = 4$, $\boldsymbol{f}(\boldsymbol{w}) = \boldsymbol{w}^3$

$\boldsymbol{w} = y_1\Phi(\boldsymbol{x}_1) + y_2\Phi(\boldsymbol{x}_2)$

follow pattern $(a+b)^3 = a^3 + 3a^2b + 3ab^2 + b^3$

$\boldsymbol{f}(\boldsymbol{w}) =$

$y_1{}^3\Phi^3(\boldsymbol{x}_1) + 3y_1{}^2y_2\Phi^2(\boldsymbol{x}_1)\Phi(\boldsymbol{x}_2) + 3y_1y_2{}^2\Phi(\boldsymbol{x}_1)\Phi^2(\boldsymbol{x}_2) +$

$y_2{}^3\Phi^3(\boldsymbol{x}_2) =$

$y_1\Phi(\boldsymbol{x}_1^3) + 3y_2\Phi(\boldsymbol{x}_1^2)\Phi(\boldsymbol{x}_2) + 3y_1\Phi(\boldsymbol{x}_1)\Phi(\boldsymbol{x}_2^2) + y_2\Phi(\boldsymbol{x}_2^3) =$

$y_1\Phi(\boldsymbol{x}_1^3) + 3y_2\Phi(\boldsymbol{x}_1^2\boldsymbol{x}_2) + 3y_1\Phi(\boldsymbol{x}_1\boldsymbol{x}_2^2) + y_2\Phi(\boldsymbol{x}_2^3)$

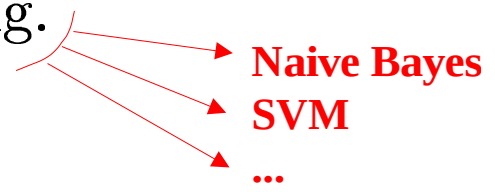
Then p-norm margin $\boldsymbol{f}(\boldsymbol{w})^\top\Phi(\boldsymbol{x}) =$

$y_1K(\underbrace{\boldsymbol{x}_1^3}_{SV},\boldsymbol{x}) + 3y_2K(\underbrace{\boldsymbol{x}_1^2\boldsymbol{x}_2}_{SV},\boldsymbol{x}) + 3y_1K(\underbrace{\boldsymbol{x}_1\boldsymbol{x}_2^2}_{SV},\boldsymbol{x}) + y_2K(\underbrace{\boldsymbol{x}_2^3}_{SV},\boldsymbol{x})$

Batch algorithm run on-line

$S = (\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_T, y_T) \in \mathbb{R}^n \times \{-1, +1\}$

A is **generic batch** classification alg.



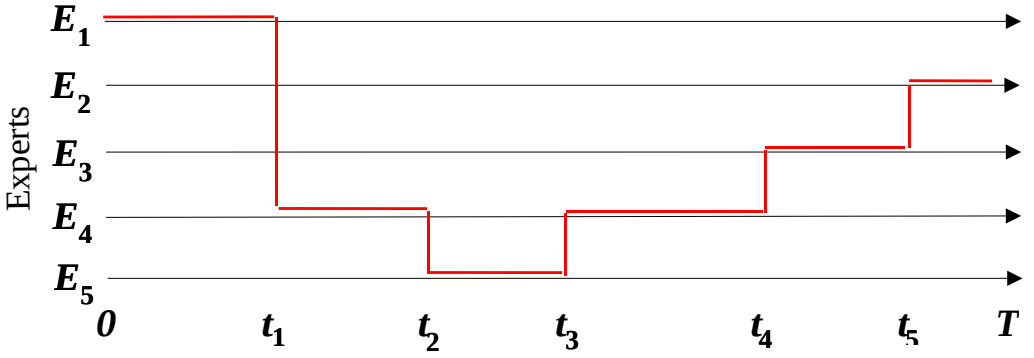
In trial t:

- train A on prefix $S_{t-1} = (\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_{t-1}, y_{t-1})$
- Get $h_t(\boldsymbol{x}) = A_{S_{t-1}}(\boldsymbol{x})$
- Mistake if $h_t(\boldsymbol{x}_t) \neq y_t$

Can count # mistakes on sequence S

Tracking the target:
Motivation and relative loss [LW94,HW98,AW98]

Previous methods yield good results when *single* target (expert/vector) is good on whole sequence S



Non-stationary tasks:
good targets
change with time

Given $S = (\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_T, y_T)$, want to bound

$$L_A(S) - a \min_{\boldsymbol{u}_1, \boldsymbol{u}_2, ..., \boldsymbol{u}_T} \underbrace{\left(Loss_{\boldsymbol{u}_0, \boldsymbol{u}_1, ..., \boldsymbol{u}_T}(S) + \sum_{t=1}^{T} dist(\boldsymbol{u}_t, \boldsymbol{u}_{t+1}) \right)}_{\text{loss of "shifting" target}}$$

Tracking the target: Examples

- Experts framework:

$\boldsymbol{u}_t$ are the n unit vectors, $\quad \boldsymbol{x}_t \in [-1,+1]^n$

$$Loss_{\boldsymbol{u}_0,\boldsymbol{u}_1,\ldots,\boldsymbol{u}_T}(S) \;=\; \sum_{t=1}^{T} \tfrac{1}{2}|\boldsymbol{u}_t{}^{\top}\boldsymbol{x}_t - y_t| \qquad \text{(mistakes)}$$

$$dist(\boldsymbol{u}_t,\boldsymbol{u}_{t+1}) \;=\; ||\boldsymbol{u}_{t+1}-\boldsymbol{u}_t|| = \begin{cases} 1 & \text{if } \boldsymbol{u}_{t+1} \neq \boldsymbol{u}_t \\ 0 & \text{otherw.} \end{cases} \qquad \text{(shifts)}$$

- Linear-threshold functions:

$\boldsymbol{u}_t,\ \boldsymbol{x}_t \in \mathbb{R}^n$

$$Loss_{\boldsymbol{u}_0,\boldsymbol{u}_1,\ldots,\boldsymbol{u}_T}(S) \;=\; \sum_{t=1}^{T} \max\{0, 1 - y_t\boldsymbol{u}_t{}^{\top}\boldsymbol{x}_t/\gamma\}$$

$$dist(\boldsymbol{u}_t,\boldsymbol{u}_{t+1}) \;=\; ||\boldsymbol{u}_{t+1}-\boldsymbol{u}_t||$$

Tracking the target: Algorithms/1

- Experts framework:

Basic expert algs. (exponential update) prevent experts to recover when they start performing better

Modification to Weighted Majority (fixed share alg.):

- $w'_{t,i} = \beta^{L_{i,t+1}}$
- Each expert sends fraction $\frac{\alpha}{n-1}$, $\alpha > 0$, of total weight to other $n-1$ experts $\rightarrow \boldsymbol{w}_{t+1}$
- $v_{t+1,i} = w_{t+1,i} / \sum_{j=1}^{n} w_{t+1,j}$
- $\widehat{y}_{t+1} = \boldsymbol{v}_{t+1}^{\top} \boldsymbol{x}_{t+1}, \quad x_{t+1,i} \in [-1, +1]$

Tracking the target: Algorithms/2

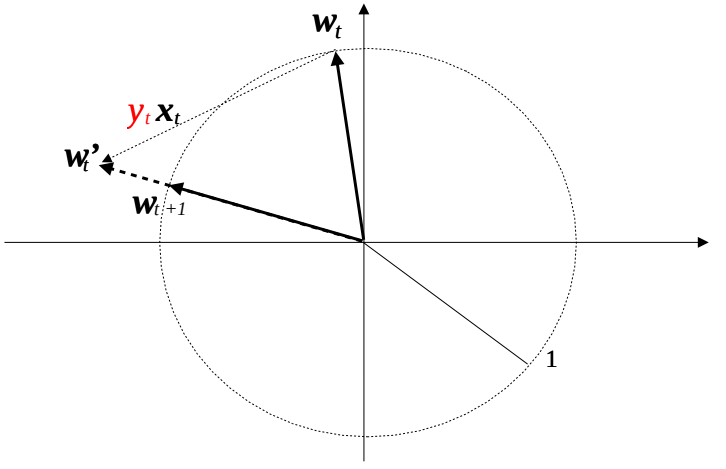
- Linear-threshold functions:

Same story ...

Modification to linear-threshold algs. (projection-based)

Simplest example:

- $\boldsymbol{w}'_t = \boldsymbol{w}_t + \eta\, y_t \boldsymbol{x}_t$
- *projects* $\boldsymbol{w}'_t$ onto
  convex set (e.g. unit ball)
  $\rightarrow \boldsymbol{w}_{t+1}$
- $\widehat{y}_{t+1} = \mathrm{SGN}(\boldsymbol{w}_{t+1}^{\top} \boldsymbol{x}_{t+1})$



Tracking the target: Bounds [HW98,AW98,...]

- Experts framework:

Bits to encode shift times

Bits to encode experts active at each shift

A is share update alg. with $\alpha = \frac{k}{T-1}$

$$L_A(S) \leq a \min_{P(k)} L_{P(k)}(S) + b\Big(k \log(T/K) + k \log n\Big)$$

- Linear-threshold functions:

Simple case: compare to sequence of linear separators

$\boldsymbol{u}_1, ..., \boldsymbol{u}_T$: $0 < \gamma \leq y_s \boldsymbol{u}_t^\top \boldsymbol{x}_s \quad s, t = 1, ..., T$

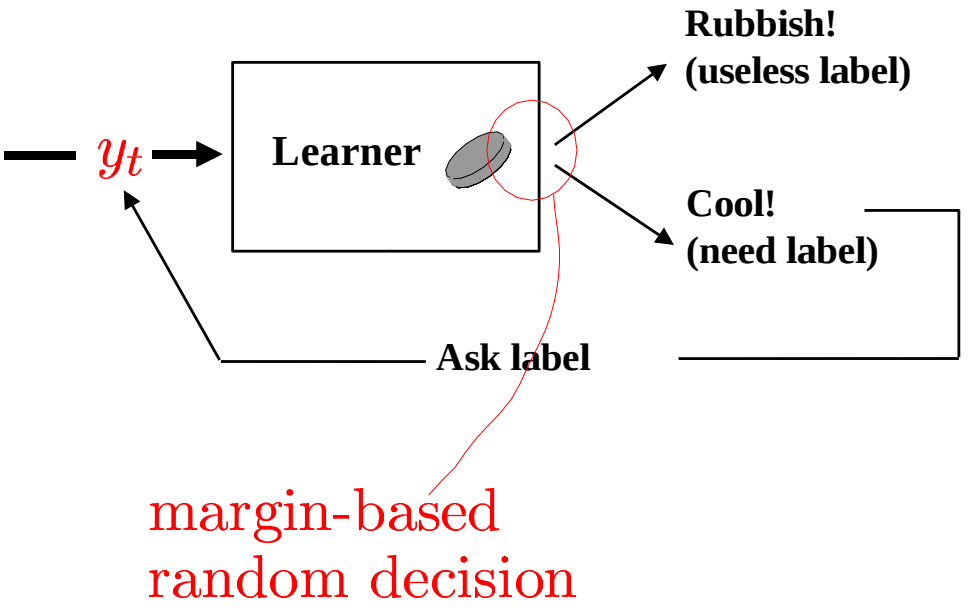
A is projection-based Perceptron with $\eta = \frac{\gamma}{2R^2}$, $||\boldsymbol{x}_t|| \leq R$

$$\#\text{ of mistakes} \leq \frac{4\,R^2}{\gamma^2}\left(\frac{1}{2} + \sum_{t=1}^{T} ||\boldsymbol{u}_{t+1} - \boldsymbol{u}_t||\right)$$

Can be generalized in several ways

Label-efficient prediction [ACL90,FSST97,CGZ04...]

Labels are scarse and/or expensive



Learner decides in on-line fashion which labels are needed:

The **higher** the margin the **less** likely label y_t is requested

Label-efficient prediction: Algorithms

Simplest example: Label-efficient Perceptron

Keep weight vector $\boldsymbol{w}_t \in \mathbb{R}^n$ and margin $r_t \in \mathbb{R}$

In trial t:

- Get instance $\boldsymbol{x}_t \in \mathbb{R}^n$, set $r_t = \boldsymbol{w}_t^\top \boldsymbol{x}_t$
- Predict with $\widehat{y}_t = \text{SGN}(r_t) \in \{-1, 1\}$
- Draw Bernoulli random variable $Z_t \in \{0, 1\}$ of bias $\frac{b}{b+|r_t|}$

positive parameter

- **If** $Z_t = 1$ **then**
  - Ask for label $y_t \in \{-1, 1\}$
  - if mistake ($y_t r_t \leq 0$) then update $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + y_t \boldsymbol{x}_t$

  Else ($Z_t = 0$) $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t$

Label-efficient prediction: Bounds

Simplest result:

When S is separated by $\boldsymbol{u}$: $||\boldsymbol{u}||_2 = 1$ with margin $\gamma \leq y_t \boldsymbol{u}^\top \boldsymbol{x}_t \ \forall t$ and $||\boldsymbol{x}_t|| \leq R$ gets

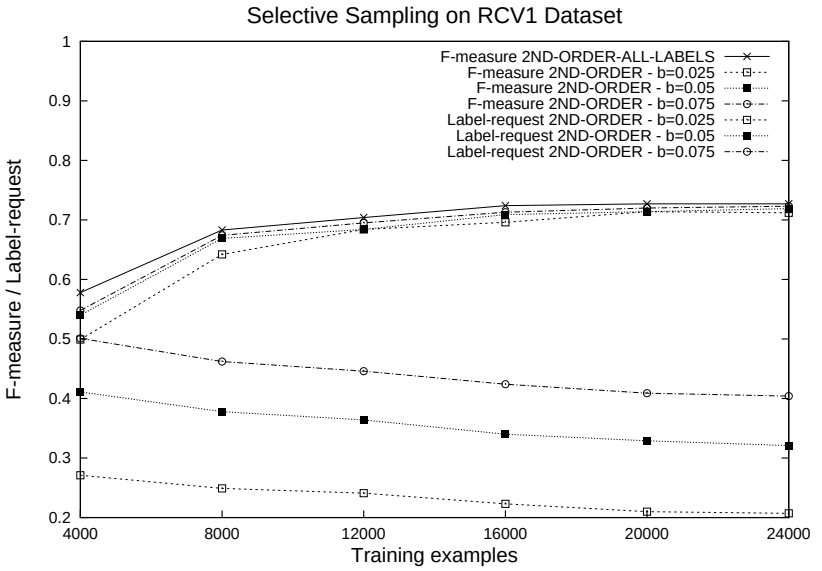
$$\text{Expected } \# \text{ of labels} = \sum_{t=1}^{T} \mathbb{E}\left[\frac{b}{b + |r_t|}\right] \leq T,$$

and, for $b = R^2/2$,

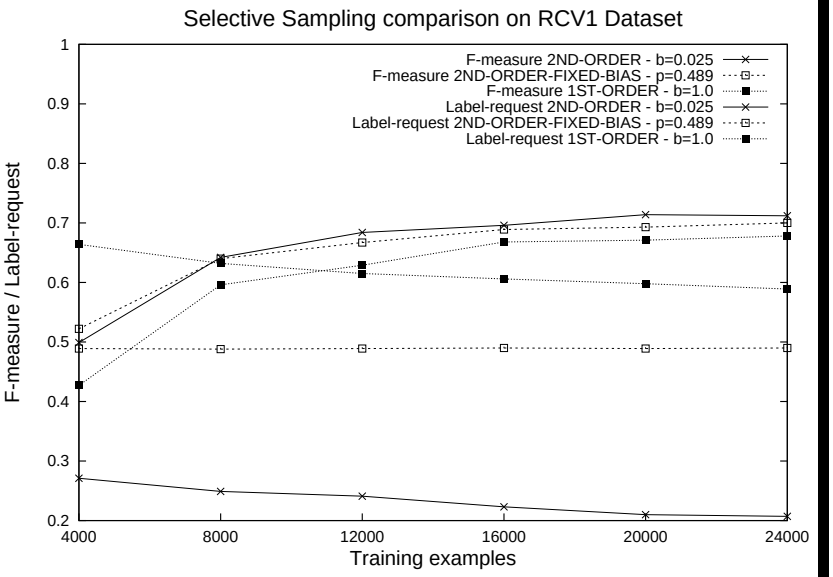
$$\text{Expected } \# \text{ of mistakes} \leq \frac{R^2}{\gamma^2}$$

Can be generalized in several ways

Label-efficient prediction: Empirical evidence/1

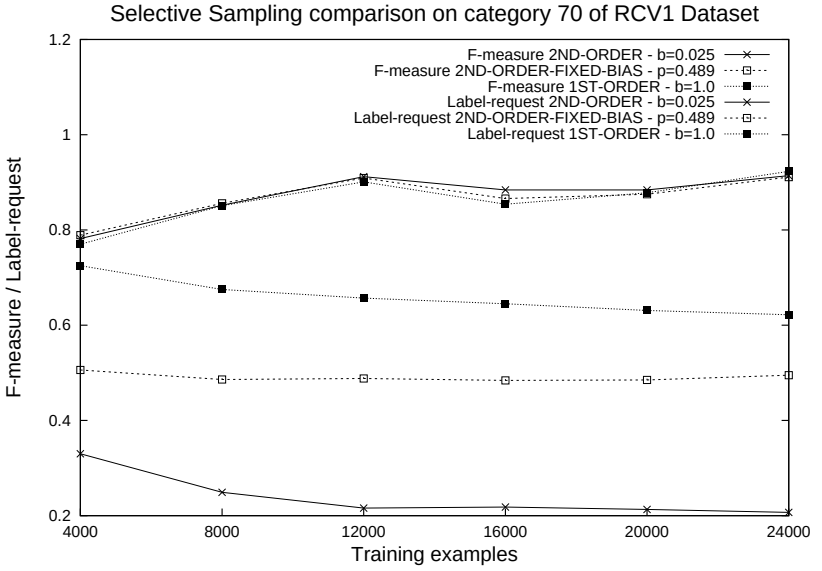


(a)



(b)

Label-efficient prediction: Empirical evidence/2



(c)

(d)

End of Part 1

Generalization bounds/1

Given

- class $\mathcal{H}$ of ± 1 functions
- i.i.d. sequence $S = (X_1, Y_1), ..., (X_T, Y_T)$ over $\mathbb{R}^n \times \{-1, 1\}$,

0−1 loss
in our case

want to compute hypothesis $\widehat{H} = \widehat{H}_S$ with small risk
$\text{risk}(\widehat{H}) = \mathbb{E}_{X,Y}[loss(Y, \widehat{H}(X))]$:

$$\mathbb{P}\left(\text{risk}(\widehat{H}) \leq \inf_{h \in \mathcal{H}} \text{risk}(h) + \epsilon\right) \geq 1 - \delta$$

Generalization bounds/2: VC Uniform conv. [VC71]

Key quantity is empirical risk

$$\text{risk}_{\text{emp}}(h) = \frac{1}{T}\sum_{t=1}^{T} loss(Y_t, h(X_t))$$

VC-bound: [VC71,Va98, ...]

constant VC−dim(*H*)

$$\mathbb{P}\left(\sup_{h\in\mathcal{H}} |\text{risk}_{\text{emp}}(h) - \text{risk}(h)| \geq c\sqrt{\frac{d + \ln 1/\delta}{T}}\right) \leq \delta$$

$\Longrightarrow \quad \widehat{H} = \text{arginf}_{h\in\mathcal{H}}\text{risk}_{\text{emp}}(h)$ is s.t.

$$\mathbb{P}\left(\text{risk}(\widehat{H}) \leq \inf_{h\in\mathcal{H}} \text{risk}(h) + 2c\sqrt{\frac{d + \ln 2/\delta}{T}}\right) \geq 1 - \delta$$

Generalization bounds/3:
Data-dep. uniform conv./1

[Ba98,BLM00,WSTSS99,BM02, ...]

$$\sqrt{\frac{d + \ln 2/\delta}{T}} \quad \rightarrow \quad C_T(S) + \sqrt{\frac{\ln 1/\delta}{T}}$$

$C_T(S) = C_T(S, \mathcal{H})$

is sample statistic:

- empirical VC-entropy [BLM00,WSTSS99]
- Rademacher complexity[BM02]
- Maximum discrepancy[BLM00]
- ...

Stronger than VC since $C_T(S) \approx \mathbb{E}[C_T(S)] << \sqrt{d/T}$

Generalization bounds/3:
Data-dep. uniform conv./2

Others (e.g., margin-based bounds for linear-threshold functions) [AKLL02,KP02,LSM01,SFBL98, ...]

$$\mathbb{P}\left(\forall h \in \mathcal{H} : \mathrm{risk}(h) \leq \mathrm{risk}_{\mathrm{emp}}(h) + C_T(h,S) + c\sqrt{\frac{\ln 1/\delta}{T}}\right) \geq 1-\delta$$

Leave algorithmic problem of computing $h \in \mathcal{H}$ optimizing trade-off

$$\mathrm{risk}_{\mathrm{emp}}(h) \quad \text{vs} \quad C_T(h,S)$$

Digression: martingales/1

Coin-tossing game:

$X_1, X_2, ..., X_t$ are ± 1 i.i.d. variables with $\mathbb{P}(X_t = +1) = 1/2$

Gambler's strategy:

$L_t = L_t(X_1, X_2, ..., X_t)$ is gain (or loss) at time t

$L_1, L_2, ..., L_t$ are **no longer** independent

Game is *fair* if

$$\mathbb{E}[L_{t+1} \,|\, X_1, ..., X_t] = 0 \quad \text{(w.p.1)} \quad \forall t$$

- Sequence $L_1, L_2, ...,$ is *martingale difference sequence* (w.r.t. $X_1, X_2, ...,$)
- Partial sums $S_t = L_1 + L_2 + ... + L_t$ is *martingale* (w.r.t. $X_1, X_2, ...,$)

Digression: martingales/2

Laws of large numbers
(empirical average concentrates around mean)
extend to martingales

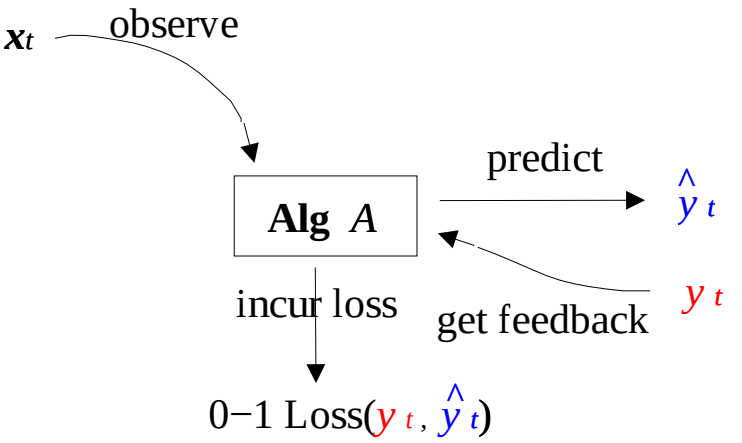
Independent variables	Dependent variables
(Zero-mean) independent r.v.	Martingale diff. sequence
Sum of (zero-mean) indep. r.v.	Martingale

(Hoeffding-Azuma) If $L_1 + L_2 + ... + L_T$ is martingale difference sequence with bounded L_t

$$\frac{L_1 + L_2 + ... + L_T}{T} \approx 0 \quad \text{(with high probability)}$$

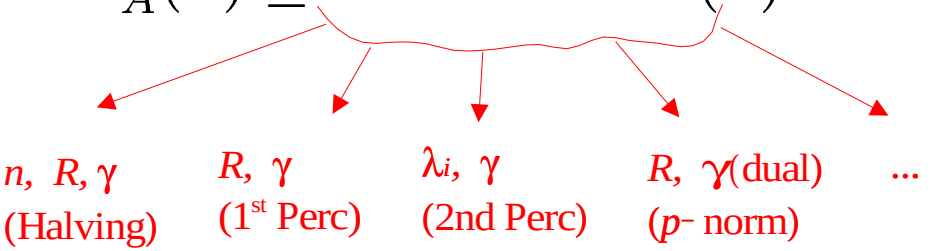
On-line pointwise bounds

$S = (\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_T, y_T)$

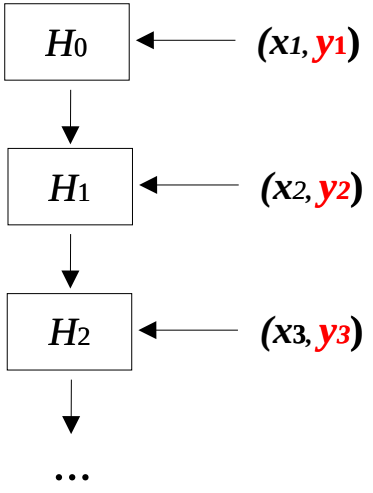


Pointwise bounds so far:

$$\text{Total \# mistakes}_A(S) \leq \text{some\_function}(S)$$



On-line pointwise $\rightarrow$ i.i.d. data-dependent/1



Sweep through sequence of examples S just once!

Get sequence of hypotheses

$H_0, H_1, H_2, ..., H_T$: $H_t = H_t((\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_t, y_t))$

Goal: Extract one with small risk

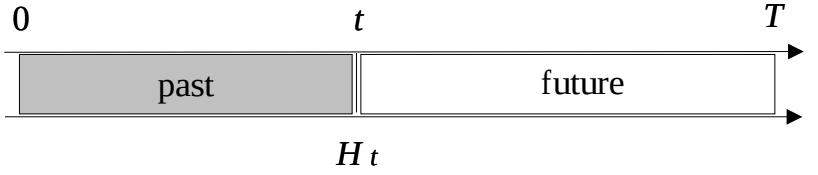
Early ref: [L] (separate test set)

On-line pointwise → i.i.d. data-dependent/2

Which one?

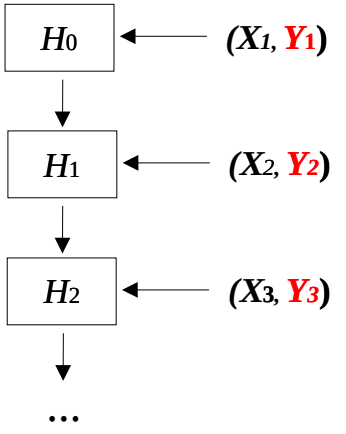
1. Last one: H_T (back to uniform convergence ...)
2. Average one: $\overline{H} = \frac{1}{T}\sum_{t=0}^{T} H_t \in [0,1]$
   (convex upper bound on 0-1 loss)
3. Best penalized one:

$$\text{risk}_{\text{emp}}(H_t, t+1) = \frac{1}{T-t}\sum_{i=t+1}^{T} loss(Y_i, H_t(X_i))$$



$$\widehat{H} = \underset{t=0\ldots T-1}{\text{argmin}} \left(\text{risk}_{\text{emp}}(H_t, t+1) + \underbrace{\sqrt{\frac{1}{T-t}\ln\frac{T}{\delta}}}_{\text{penalty}} \right)$$

On-line pointwise → i.i.d. data-dependent/3
Proof technique/1



H_0

$H_1 = H_1((X_1, Y_1))$

$H_2 = H_2((X_1, Y_1), (X_2, Y_2))$

...

Build martingale kind of process

$$L_t = L_t((X_1, Y_1), ..., (X_t, Y_t))$$
$$= loss(Y_t, H_{t-1}(X_t)) - \text{risk}(H_{t-1})$$

$$\mathbb{E}[loss(Y_t, H_{t-1}(X_t)) \mid (X_1, Y_1), ..., (X_{t-1}, Y_{t-1})]$$

On-line pointwise $\rightarrow$ i.i.d. data-dependent/3
Proof technique/2

From the very definition of L_t

$L_1, L_2, ..., L_T$ is bounded ($|L_t| \leq 1$)

martingale difference sequence

w.r.t. $(X_1, Y_1), ..., (X_T, Y_T)$

Hoeffding-Azuma:

$$\frac{1}{T} \sum_{t=1}^{T} \Big[loss(Y_t, H_{t-1}(X_t)) - \text{risk}(H_{t-1}) \Big] \approx 0$$

On-line pointwise → i.i.d. data-dependent/3
Proof technique/3

$$\overbrace{\underbrace{\frac{1}{T}\sum_{t=1}^{T} loss(Y_t, H_{t-1}(X_t))}_{\#\text{ of mistakes}}}^{M_T} \underbrace{\approx}_{(*)} \frac{1}{T}\sum_{t=1}^{T} \text{risk}(H_{t-1}) \begin{cases} \underbrace{\geq}_{(**)} \text{risk}(\overline{H}) \\ \underbrace{\approx}_{(***)} \text{risk}(\widehat{H}) \end{cases}$$

(*) (Hoeffding-Azuma) [DGL96]

(**) bounded and convex (Jensen)

(***) general bounded (Chernoff-Hoeffding) [DGL96]

On-line pointwise → i.i.d. data-dependent/4
Simplest bounds

Convex: $$\mathbb{P}\left(\text{risk}(\overline{H}) \le M_T + L\sqrt{\frac{2}{T}\ln\frac{2}{\delta}}\right) \ge 1-\delta$$

bound on range of convex loss

More general: $$\mathbb{P}\left(\text{risk}(\widehat{H}) \le M_T + 6\sqrt{\frac{1}{T}\ln\frac{T}{\delta}}\right) \ge 1-\delta$$

On-line pointwise $\rightarrow$ i.i.d. data-dependent/5

Some applications: plug and play/1

Recall bound on Halving Algorithm for separable case:

$$M_T \leq \frac{1}{T} O\left(d \log(R/\gamma)\right)$$

Just plug back into

$$\mathbb{P}\left(\text{risk}(\widehat{H}) \leq M_T + 6\sqrt{\frac{1}{T}\ln\frac{T}{\delta}}\right) \geq 1-\delta$$

Gets

$$\mathbb{P}\left(\text{risk}(\widehat{H}) \leq \frac{1}{T} O\left(n \log(R/\gamma)\right) + 6\sqrt{\frac{1}{T}\ln\frac{T}{\delta}}\right) \geq 1-\delta$$

Similar to [HG02]

On-line pointwise → i.i.d. data-dependent/5

Some applications: plug and play/2

Recall bound on Kernel Perceptron:

$$M_T \leq \inf_{\gamma>0,\ f\in H_K, \|f\|=1} \frac{1}{T}\left(D_\gamma(f;S) + \frac{\sqrt{\sum_{t\in\mathcal{M}} K(\boldsymbol{x}_t,\boldsymbol{x}_t)}}{\gamma}\right)$$

Separable case:

$$M_T \leq \frac{1}{T}\frac{\max_{t\in\mathcal{M}} K(\boldsymbol{x}_t,\boldsymbol{x}_t)}{\gamma^2}$$

Plug back into

$$\mathbb{P}\left(\mathrm{risk}(\widehat{H}) \leq M_T + 6\sqrt{\frac{1}{T}\ln\frac{T}{\delta}}\right) \geq 1-\delta$$

Similar to [BM02] for SVM

On-line pointwise → i.i.d. data-dependent/5

Some applications: plug and play/3

Recall bound on Kernel Second-order Perceptron (separable case)

$$M_T \leq \frac{1}{T} \frac{a + \sum_i \ln(1 + \frac{\lambda_i}{a})}{\gamma},$$

Plug into

$$\mathbb{P}\left(\text{risk}(\widehat{H}) \leq M_T + 6\sqrt{\frac{1}{T} \ln \frac{T}{\delta}} \right) \geq 1 - \delta$$

Similar to [WSTSS99] for SVM

On-line pointwise → i.i.d. data-dependent/5

Some applications: plug and play/4

Recall bound on Perceptron for shifting targets

$$M_T \leq \frac{1}{T} \frac{4\,R^2}{\gamma^2} \left(\tfrac{1}{2} + \sum_{t=1}^{T} ||\boldsymbol{u}_{t+1} - \boldsymbol{u}_t|| \right)$$

Again, combine with

$$\mathbb{P}\left(\text{risk}(\widehat{H}) \leq M_T + 6\sqrt{\frac{1}{T} \ln \frac{T}{\delta}} \right) \geq 1 - \delta$$

On-line pointwise $\to$ i.i.d. data-dependent/5

Some applications: plug and play/5

Recall bounds on label-efficient Perceptron:

$$\mathbb{E}_{(*)}[M_T] \quad \leq \quad \frac{1}{T}\,\frac{R^2}{\gamma^2}$$

$$\mathbb{E}_{(*)}[\#\text{ of labels}] \quad = \quad \sum_{t=1}^{T} \mathbb{E}_{(*)}\left[\frac{b}{b+|r_t|}\right]$$

$(*)$ = internal randomization

Get with prob $> 1-\delta$:

$$\mathbb{E}_{(*)}[\text{risk}(\widehat{H})] \leq \frac{1}{T}\,\frac{R^2}{\gamma^2} + 6\sqrt{\frac{1}{T}\ln\frac{T}{\delta}}$$

and

On-line pointwise $\to$ i.i.d. data-dependent/5

Some applications: plug and play/6

A is your favourite batch classification alg.

Run it in on-line fashion and count # of mistakes

Still get

$$\mathbb{P}\left(\text{risk}(\widehat{H}) \leq M_T + 6\sqrt{\frac{1}{T}\ln\frac{T}{\delta}}\right) \geq 1-\delta$$

No direct mention of, e.g., "complexity of function classes"

Try it yourself with other specific algs.

On-line pointwise → i.i.d. data-dependent/6 Remarks

These bounds:

- are algorithm-specific (NO uniform convergence arguments, closer in spirit to algorithmic stability/luckiness) [BE02,HeWi02,...]
- proven by simple large deviation on martingales
- refer to efficient algs (on-line, one sweep)
- are tight (I believe ...)
- Are widely applicable (in principle)

On-line pointwise $\rightarrow$ i.i.d. data-dependent/7 Refinements/1

$$\mathbb{P}\left(\text{risk}(\widehat{H}) \leq \min_{t=0\ldots T-1}\left(M_{t,T} + 6\sqrt{\frac{1}{T-t}\ln\frac{T}{\delta}}\right)\right) \geq 1-\delta,$$

where $M_{t,T} = \frac{1}{T-t}\sum_{i=t+1}^{T} loss(Y_i, H_{i-1}(X_i))$ (loss on suffix)

Basically "$\min_{t=0\ldots T-1}$" replaces "$t = 0$"

On-line pointwise → i.i.d. data-dependent/7
Refinements/2

$$\mathbb{P}\left(\mathrm{risk}(\widehat{H}) \le M_T + O\left(\frac{1}{T}\ln\frac{T}{\delta} + \sqrt{\frac{M_T}{T}\ln\frac{T}{\delta}}\right)\right) \ge 1-\delta,$$

$$\widehat{H} = \operatorname*{argmin}_{t=0\ldots T-1}\left(\mathrm{risk}_{\mathrm{emp}} + \underbrace{\frac{1}{T-t}\ln\frac{T}{\delta} + \sqrt{\frac{\mathrm{risk}_{\mathrm{emp}}}{T-t}\ln\frac{T}{\delta}}}_{\text{penalty}}\right)$$

$$\mathrm{risk}_{\mathrm{emp}} = \mathrm{risk}_{\mathrm{emp}}(H_t, t+1) = \frac{1}{T-t}\sum_{i=t+1}^{T} loss(Y_i, H_t(X_i))$$

(Uses Bernstein-type inequalities for martingales) [F75,DZ01]

Can be combined with "Refinement/1"

Conclusions

- Pointwise bounds for on-line algorithms directly turn to (tight) data-dependent i.i.d. bounds
- Easy plug and play
- Resulting algs. are still as efficient as on-line (one epoch over training sequence)
- Simple proofs, algorithm-specific, no uniform convergence
- Can be immediately extended to regression frameworks